AFRL-IF-RS-TR-2004-57
Final Technical Report
March 2004

DYNAMIC SENSOR NETWORKS

University of Southern California at Marina Del Rey

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. H557

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-57 has been reviewed and is approved for publication

APPROVED: /s/

SCOTT S. SHYNE
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	MARCH 2004	Final Jun 99 – Feb 03

4. TITLE AND SUBTITLE
DYNAMIC SENSOR NETWORKS

5. FUNDING NUMBERS
C - F30602-99-1-0529
PE - 62301E
PR - H577
TA - 16
WU - 01

6. AUTHOR(S)
Brian Schott, Ronald Riley,
Mani Srivastava, and Igor Elgorriaga

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California
Information Science Institute
4676 Admiralty Way
Marina Del Rey California 90292-6695

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFGA
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2004-57

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Scott S. Shyne/IFGA/(315) 330-4819/ Scott.Shyne@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
The DSN team investigated hardware and software technologies for unattended ground sensor applications. Task 1: Distribution and Aggregation includes node localization techniques, low-power data link protocols, power aware routing protocols, and spatial addressing and routing. Task 2: Declarative Languages and Execution Environment includes topographical soldier interface and a sensor network simulation environment for algorithm development, deployment planning, and operational support. Finally, Task 3: Platforms include investigative hardware development to support laboratory communications, processing, or localization experiments as well as open-source COTS PDA integration and system software in support of the soldier interface.

14. SUBJECT TERMS
Unattended Ground Sensors, Wireless Networking, Embedded Linux

15. NUMBER OF PAGES
243

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The Dynamic Sensor Networks project investigated software tools and techniques for unattended ground sensor systems. The DARPA SensIT program introduced a sensor network concept for military and civilian uses that had nodes consisting of an embedded processor, digital radio, and a number of sensors. The nodes form ad-hoc distributed processing networks producing high-quality information with limited resource consumption. Sensor networks support quick decision-making and provide timely and accurate situation awareness for soldiers in the field (via a PDA-like device) and remote operators in a command center. An artist system concept is shown in Figure 1. This research was intended to advance this system concept.

Figure 1: DSN Systems Concept

The DSN project organized a multidisciplinary research team. The USC Information Sciences Institute and Distant Focus Inc. researched platform technologies, communications subsystems, embedded operating systems, and handheld display systems. Virginia Tech concentrated on declarative languages and execution environments, including status visualization and planning tools. UCLA developed distribution and aggregation technologies such as network protocols and network simulation tools. The DSN project was divided into three tasks:

- ***Task 1: Distribution and Aggregation*** includes node localization techniques, low-power data link protocols, power aware routing protocols, and spatial addressing and routing.
- ***Task 2: Declarative Languages and Execution Environment*** includes topographical soldier interfaces and a sensor network simulation environment for algorithm development, deployment planning, and operational support.
- ***Task 3: Platforms*** includes investigative hardware development to support laboratory communications, processing, or localization experiments as well as open-source COTS PDA integration and system software in support of the solider interface.

For the purpose of providing focus for this research, the DSN team defined three concept operation (CONOP) scenarios. The first CONOP is *intelligence gathering*; small-scale collaboration where sensor nodes form small ad hoc groups to identify and track targets to provide human-in-the-loop fire support. The second is called *collaborative consensus*. This is intended to provide reliable data from a large number of unreliable nodes such as from chem/bio sensors. The third CONOP is health, status, plus *augmented awareness*. This scenario introduces the idea of friendly forces traveling through a sensor net gaining tactical support from local sensor nodes. Combined, these three scenarios defined distinct modes of operation of a sensor network and operational requirements that drive the architecture definition. The SensIT program sponsored a series of field experiments to stimulate the development of collaborative technologies and facilitate validation of the multiple approaches.

The following sections are organized by task. Section 2 details the efforts related to *Task 1: Distribution and Aggregation*, Section 3 covers *Task 2: Declarative Languages and Execution Environments*, and Section 4 discusses *Task 3: Platforms*. Section 5 covers Field Experiments. Section 6 covers a summary of all deliverables. Section 7 presents the personnel and their contributions to this effort. Section is a list of Publications resulting from this effort. Section 9 is a list of Acronyms and finally, Section 10 is a list of Addenda.

2 DISTRIBUTION AND AGGREGATION

Important characteristics of wireless sensor networks are how they communicate and collaborate to understand the environment. The physically distributed sensor nodes leverage spatial diversity to improve target classification and triangulate/track targets within a sensor field. Management of their aggregate behavior defines sensor field lifetime, capabilities, and performance. The DSN project investigated a number of topics related to the distribution and aggregation research area. The primary focus was related to the development of a modeling and simulation framework called SensorSim, which is described in Section 2.1. The simulation tools were used to develop and characterize *media access control* (MAC) and wireless network routing protocols developed by the DSN team. These efforts are described in Section 2.2. Finally, localization of nodes is important in these distributed collaborative systems. Techniques for GPS-less localization are detailed in Section 2.3.

2.1 Modeling and Simulation

2.1.1 SensorSim

UCLA developed a framework for detailed modeling and simulation of distributed sensor networks called *SensorSim*. This tool is an extended version of *ns*, an open-source network protocol simulator. The basic approach was to introduce to *ns* the notion of a set of node resources (processor, sensor, radio etc.) that are used for various tasks in the protocol stack and the application agents. Resources have a number of modes with different levels of power consumption. For example, the radio model has *transmit*, *receive*, *idle*, and *sleep* modes, and takes into account effects of data rate and radio frequency (RF) power amplification on the overall power consumption. These models are used to compare and evaluate design tradeoffs for various aspects of the distributed system, such as media access control (MAC), network routing, distributed data management, and application-level interactions. *SensorSim* introduced a number of features unique to network simulators:

- The framework adds hybrid simulation capabilities to *ns*. The simulated network can consist of a mix of virtual (simulated) nodes and real (physical) nodes. In other words, some of the nodes in the simulated network have real nodes performing the work. This is particularly useful in evaluating MAC protocols. The approach is based on using a PC with a node as a gateway to a network of real nodes, each of which is represented by a proxy node in the simulated network. The simulated network generates stimulus traffic for the smaller “real” sensor network.
- Another form of hybrid simulation capability was added to *ns* whereby an application written as a Unix process can “run” on top of a simulated node. Essentially, the application interfaces to the routing layer of the simulated node via an API that allows packets to be sent and received. This capability has been leveraged at UCLA to simulate mobile scripts developed under the SensIT Sensorware project, which was led by Rockwell Science Center.
- The battery simulation models that go beyond the naive "fixed bucket of energy.” For example, the battery model takes into account the rate at which the power is being consumed. Additional models were developed that take into account the pulse discharge profile generated by the protocols.

- Targets are modeled in the simulation framework as target node entities. They interact with sensor node entities over sensor channels that can be modeled with physical layer models or logical sensor layer (probability detection event) models.

The *SensorSim* architecture is shown in Figure 2. The user node runs a user application that interacts with the network simulation over wireless channels. A number of real and simulated sensor nodes execute the distributed sensor processing application(s) and interact with other nodes over the wireless channel(s) and targets over the sensor channel(s). The functional model of the node simulates the hardware and measures performance. The power model of the node logs power consumption of the various simulated components. Finally, target nodes simulate the behavior of targets of interest and generate stimulation datasets for the nodes. Alternatively, real data can be used from data collection experiments.

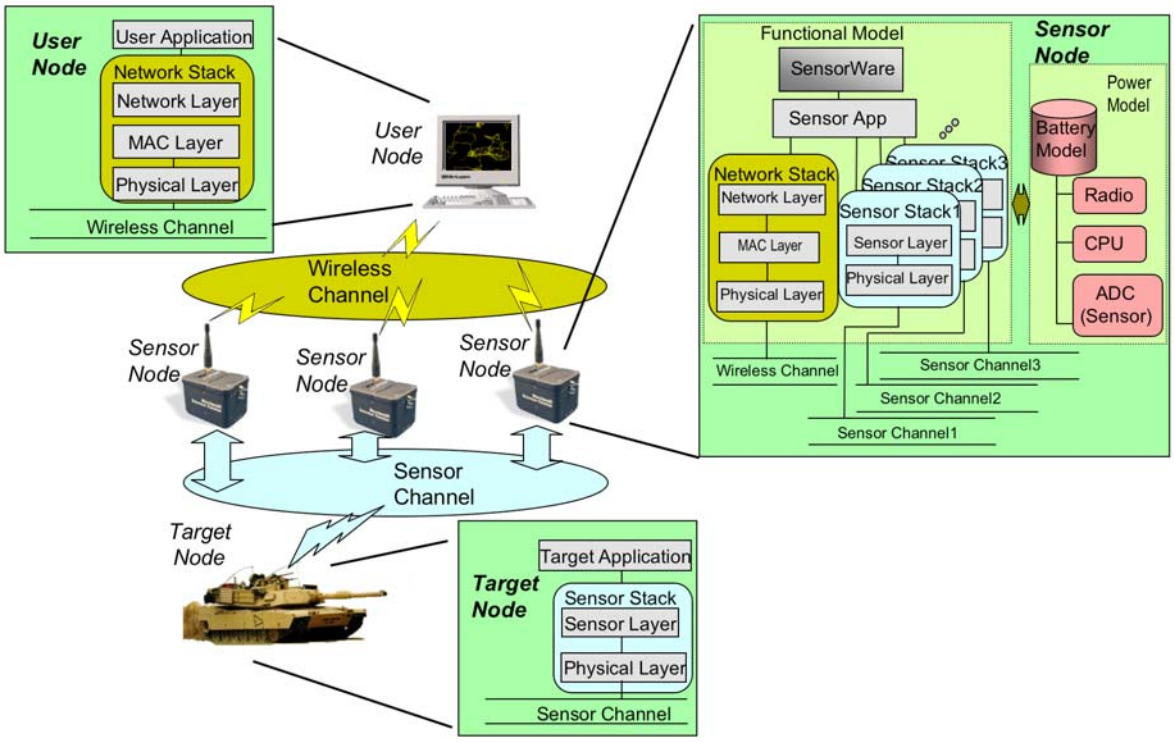


Figure 2: SensorSim Architecture

The *SensorSim* tool is available on the UCLA web site: http://nesl.ee.ucla.edu/sensorsim/. It has been used extensively in this project to evaluate sensor networking protocols and collaborative application environments and has been adopted by a number of the SensIT community researchers. Many of the features and models have been incorporated into the *ns2* sensor network simulation tool environment and are now made available to a broad network research community. The *SensorSim* tool was also used to evaluate data collections at the various SensIT field experiments.

2.1.2 SensorVis

In addition to *SensorSim*, the UCLA team developed a scenario generation and visualization tool called *SensorVis*. This tool supports diverse scenario generation, including node deployment patterns, target trajectories, sensor characteristics, and node attributes. The tool can be slaved to a

running *SensorSim* simulation to configure the experiment and visualize the results overlaid on publicly available maps. It can also be used interactively to monitor and track the activities in a real sensor network. *SensorVis* can generate XML output for further analysis of the behavior of the sensor network and has been used to read in SITEX experiment scenarios. A simulated sensor network scenario of the UCLA campus is shown in Figure 3. *SensorVis* also introduced other planning capabilities to the DSN sensor tool suite. Several coverage analysis algorithms were integrated with the scenario builder. This allows the sensor network designer to create topologies with desired coverage performance.

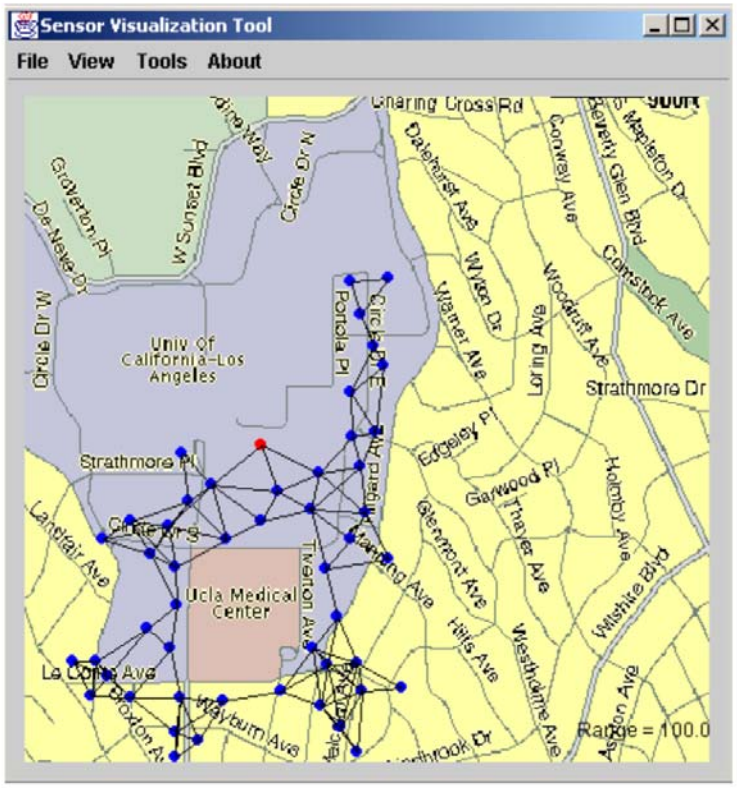


Figure 3: SensorVis UCLA Campus Scenario

2.2 Communications and Networking Protocols

2.2.1 Spatial Addressing

Since wireless spectrum is a broadcast medium, each wireless interface requires a media access control identifier (MAC address). However, the typical data payload size in wireless sensor networks is very small. A globally unique MAC address appended to every packet would present too much protocol overhead for very small packets. DSN addresses this problem by using a smaller address size and employing spatial address reuse, similar to those used in cellular systems. Figure 4 shows an example of spatial address reuse. There are two aspects of this problem: dynamic assignment algorithms, and address representation.

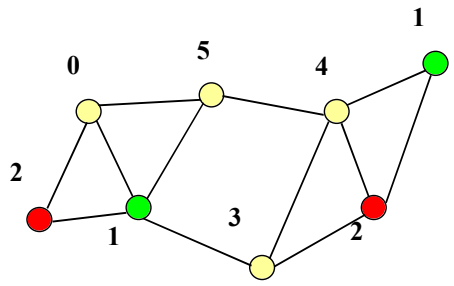


Figure 4: Spatial Address Reuse Example

The dynamic assignment algorithm developed under this effort operates as shown in Figure 5: 1) the network is operational (nodes have a valid address), 2) the nodes listen to periodic broadcasts of neighboring nodes, 3) in case of an address conflict, the node is notified, and 4) a non-conflicting address is chosen and broadcast in a periodic cycle. At this point, the new node has joined the network. This algorithm has a number of beneficial features. It supports *additive convergence* in that the network remains operational during address selection and *mapping* from a unique id to a spatially reusable address. The algorithm is also valid with unidirectional links.

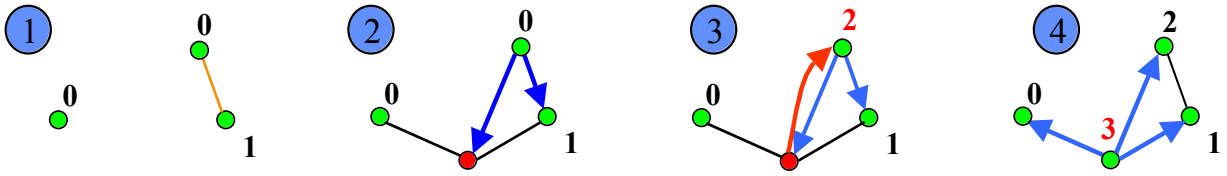


Figure 5: Distributed Assignment Algorithm

Address representation is a problem in sensor networks because the number of unique addresses varies with the node density. The UCLA approach is to use variable-length encoded MAC addresses so that the protocol dynamically picks addresses according to local node density. This approach was initially implemented and evaluated using the PARSEC simulator, then was incorporated into SensorSim. Extensive simulation and analysis has shown that the MAC header overhead is reduced by a factor of three (3X) relative to fixed addresses and by an order of magnitude (10X) relative to Ethernet-type 48-bit addresses (see Table 1). The scheme is perfectly scalable from a few to millions of nodes at practical node densities; it handles unidirectional links, and is robust to node destruction, introduction, and mobility. The scheme can be used for both TDMA-type MAC as well as CSMA-type MAC. Figure 6 shows a plot of average node density compared to average encoded address size in bits.

Table 1: Addressing Representation Schemes

Scheme	Address Selection Type	Address Size (bits)	Address Scalability
Globally Unique	Manufacturing	128	+
Network Wide Unique	Deployment	14	-
Fixed-size Dynamic	Centralized / Distributed	4.7	+/-
Encoded Dynamic	Distributed	4.4	+

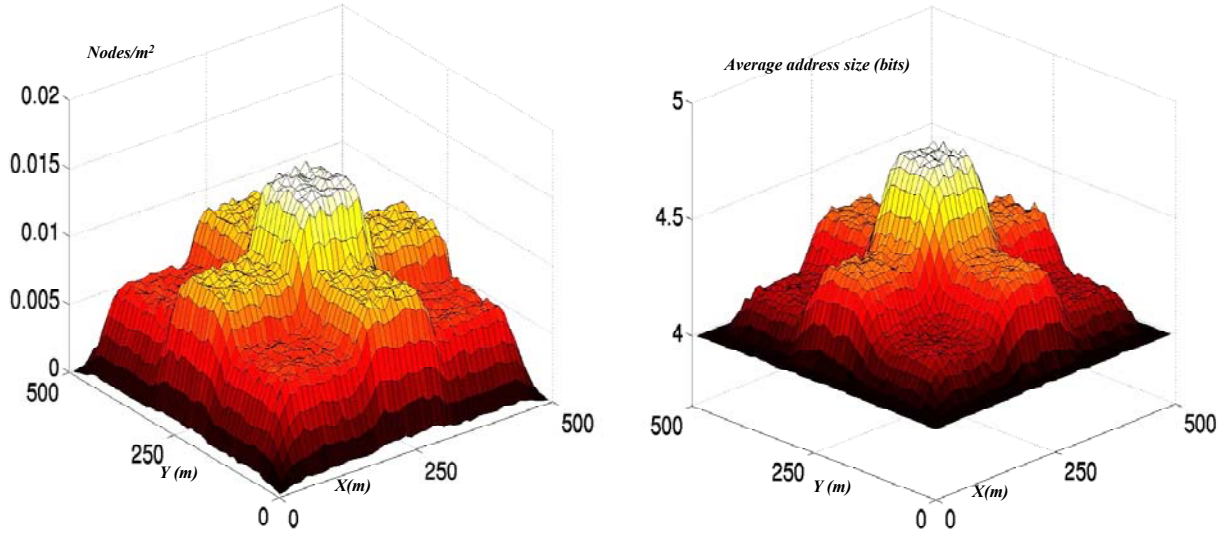


Figure 6: Node Density Versus Average Address Size

2.2.2 Low-Power Network Routing

UCLA investigated a number of low-power network routing protocol approaches that are described extensively in the publications included in Section 10. One such approach is called *Adaptive Transmission-power Heuristic and Energy-optimizing ad-hoc Network routing Algorithm* (ATHENA), which uses alternate routes to maximize network lifetime and dynamically adapts transmission power to find energy-optimal multi-hop paths. The alternate routes approach exploits the diversity of paths to distribute traffic so as to avoid burnouts along a single heavily trafficked path. Packet dispersers and combiners are used for this purpose, and several deterministic as well as stochastic algorithms for doing this were investigated via simulation in ns. The path diversity approach is applicable to routing approaches such as DSR, as well as gradient based routing. UCLA also conducted studies of what metrics are meaningful in evaluating the schemes, and proposed the use of *histogram of remaining battery energy* as an appropriate metric, with the RMS of the histogram capturing the two essential properties: area and spread. The Data Combining entities, which are created automatically at intermediate nodes, combine data packets that are headed in the same destination. It essentially lowers the area of the histogram. Simulations show a significant energy-latency trade-off is enabled by this technique relative to gradient or DSR routing. A second approach is deliberate spreading of the traffic, for which several schemes were investigated including energy-based, stochastic, stream-based, and combinations. Spreading reduces the RMS metric by reducing the spread in the histogram. Simulation results show that the stream-based scheme works best. However, spreading is not beneficial under all circumstances, and ideally it is better when averaged over all possible future traffic patterns. This requires detection of bottleneck nodes in the network.

2.3 GPS-less Localization

The discovery of absolute and relative locations of individual nodes is very important in a sensor network. The location attribute is often used in location-based naming and geographic-addressing of nodes. Location is used in geographical routing protocols. Location is also used

for tracking of moving phenomena (vehicles, personnel, etc.). The Global Positioning System (GPS) is used for this purpose in many networking systems, but it is not sufficient. GPS requires line of sight to multiple satellites, which can be hampered by trees or buildings. It also introduces cost to a sensor system and consumes power. Surrogate GPS systems have been used in military systems, but they are also susceptible to failure and add cost. The UCLA team examined approaches for deriving location if only a subset of nodes were aware of their location as shown in Figure 7.

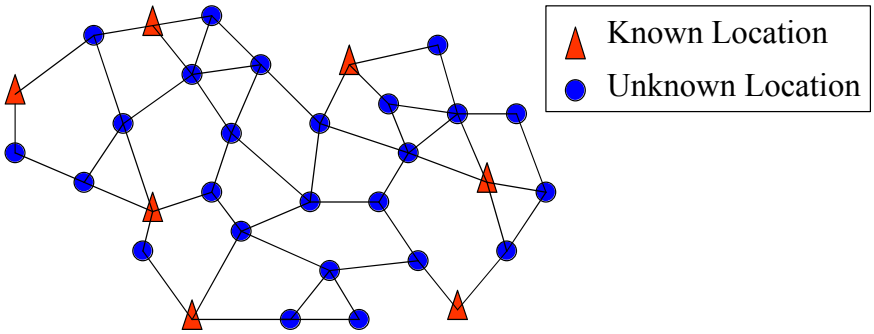


Figure 7: Sensor Network with Known and Unknown Locations

UCLA investigated a number of multilateration algorithms for GPS-less sensor node localization, including basic centralized multilateration, iterative multilateration, and finally collaborative multilateration. The algorithm for basic multilateration is given in Figure 8.

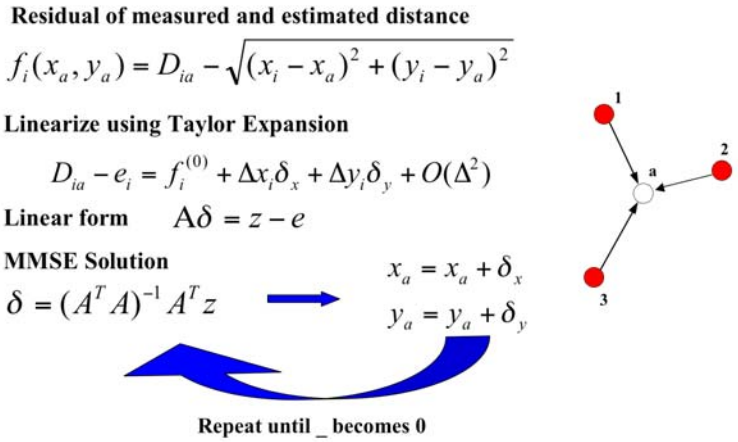


Figure 8: Basic Multilateration

In this centralized approach, the nodes route messages to a central point where the equations are solved simultaneously. This has a number of disadvantages. Timing synchronization is required and high traffic congestion around the central node leads to higher message power consumption and higher latencies for location updates. Distributed approaches are preferable. They require less traffic and therefore less power. They have better handling of environmental variations (speed of ultrasound, radio path lost), and are more robust to node failure.

Iterative multilateration, for example, is basic multilateration applied iteratively across the network. The location estimate is improved with each iteration step of the algorithm until locations settle to a steady state. This approach is diagramed in Figure 9.

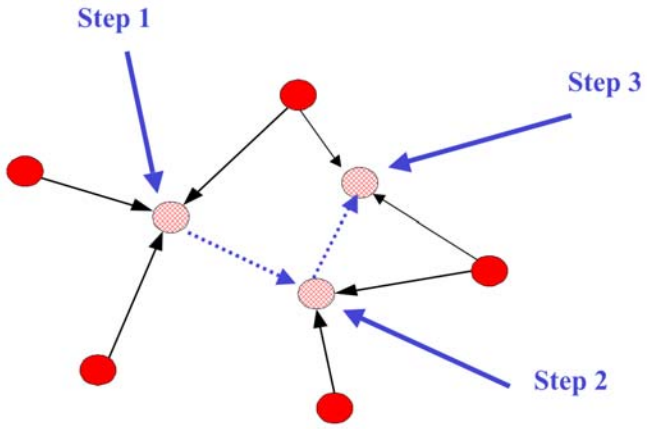


Figure 9: Iterative Multilateration

The percentage of nodes able to resolve their locations is given in Figure 10. For moderate numbers of nodes, iterative multilateration works well for percentages over 20%. However, iterative multilateration may stall if the network is very sparse or the percentage of beacons is very low. Terrain obstacles can also have a significant impact on location accuracy. One problem with iterative multilateration is that if the network is large, successive errors will accumulate and degrade location estimates.

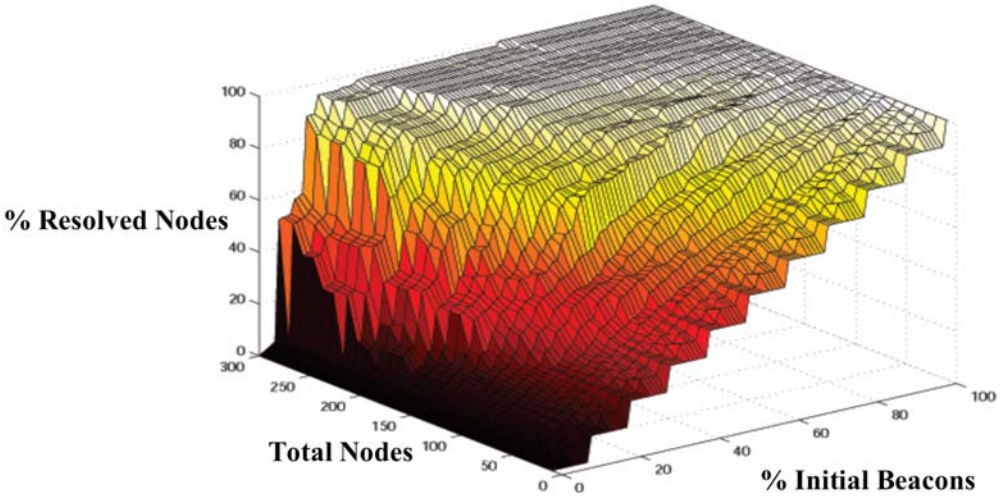


Figure 10: Resolved Nodes Versus Beacon Densities

This fact motivated the development of collaborative multilateration. Collaborative multilateration is described in Figure 11. It uses location information over multiple hops, but weights the error estimate of the indirect paths. Solving the equations operates the same as basic multilateration.

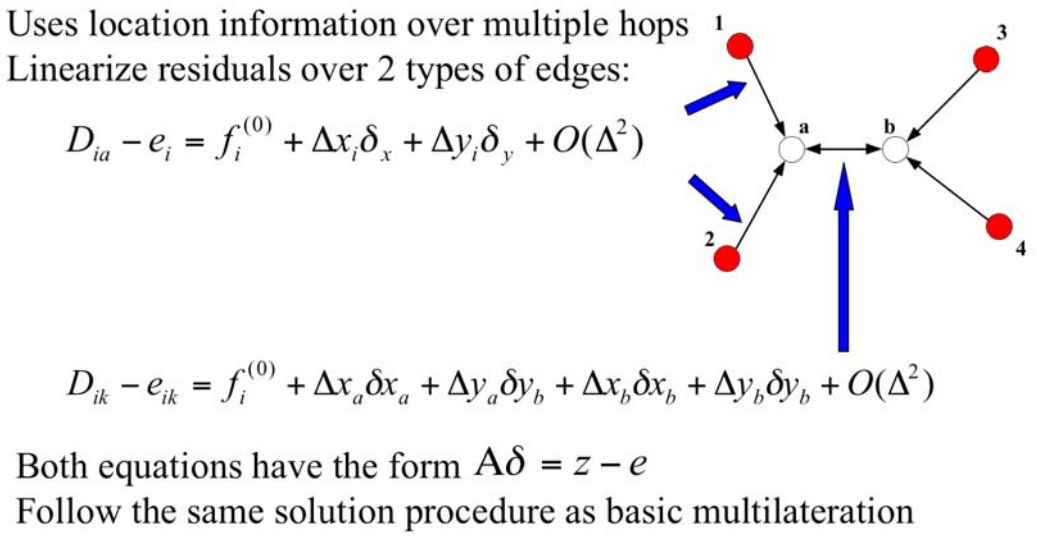


Figure 11: Collaborative Multilateration

A complementary approach uses collaborative sub-trees. With this approach, location estimation takes place at the scope of a neighborhood. Each unknown node (shown in Figure 12) must have at least three participating neighbors and a participating node is either a beacon node or an unknown node connected to three participating nodes. The collaborative sub-trees can zoom in and out to form a well-determined system. It can avoid degenerate cases and obstacles. It also reduces the error propagation. Error in the location estimates can further be reduced if computation takes place at a central point. This is less of a burden when the size of the neighborhood is bounded.

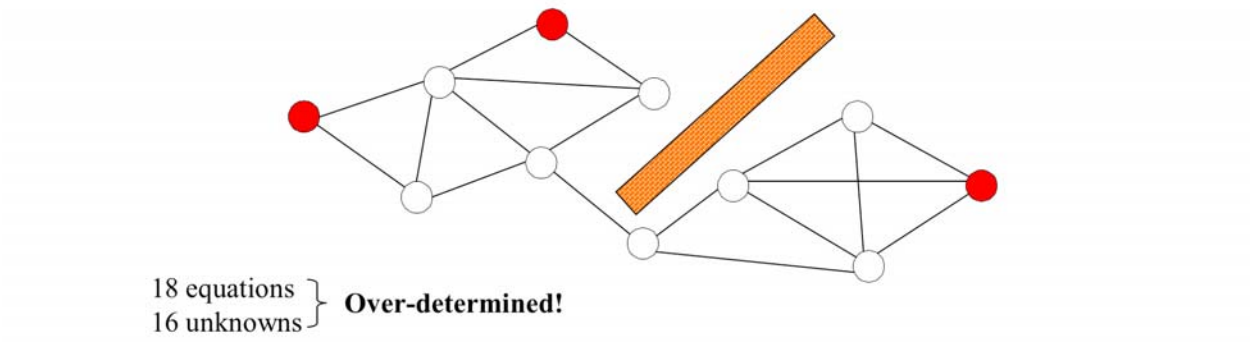


Figure 12: Collaborative Sub-Trees

Extensive simulation analysis and field-testing of these concepts was performed under this research effort. In simulation for example, the accuracy of iterative multilateration is shown in Figure 13. This analysis was performed using the SensorSim tool suite. For indoor and outdoor physical tests, two platforms were used: 1) a custom ultrasound array microsensor developed at UCLA called Medusa, and 2) commercial Rockwell HYDRA microsensors using RSSI estimates. UCLA conducted extensive RSSI measurements at various locales on UCLA campus and in places near the campus to validate the iterative multilateration based joint location and channel estimation scheme. Analysis of results indicates good localization in open areas similar to those used for SITEX experiment, but problems with multipath in the presence of nearby structures. Joint RSSI - ultrasound schemes were considered where ultrasound provides

additional distance measurement via ranging, while the radio not only provides RSSI data but also serves to synchronize the ultrasound ranging. The range accuracy characterization of these platforms is summarized in Figure 14. Further information and analysis of multilateration algorithms is provided in the academic papers included in the addendum of this report.

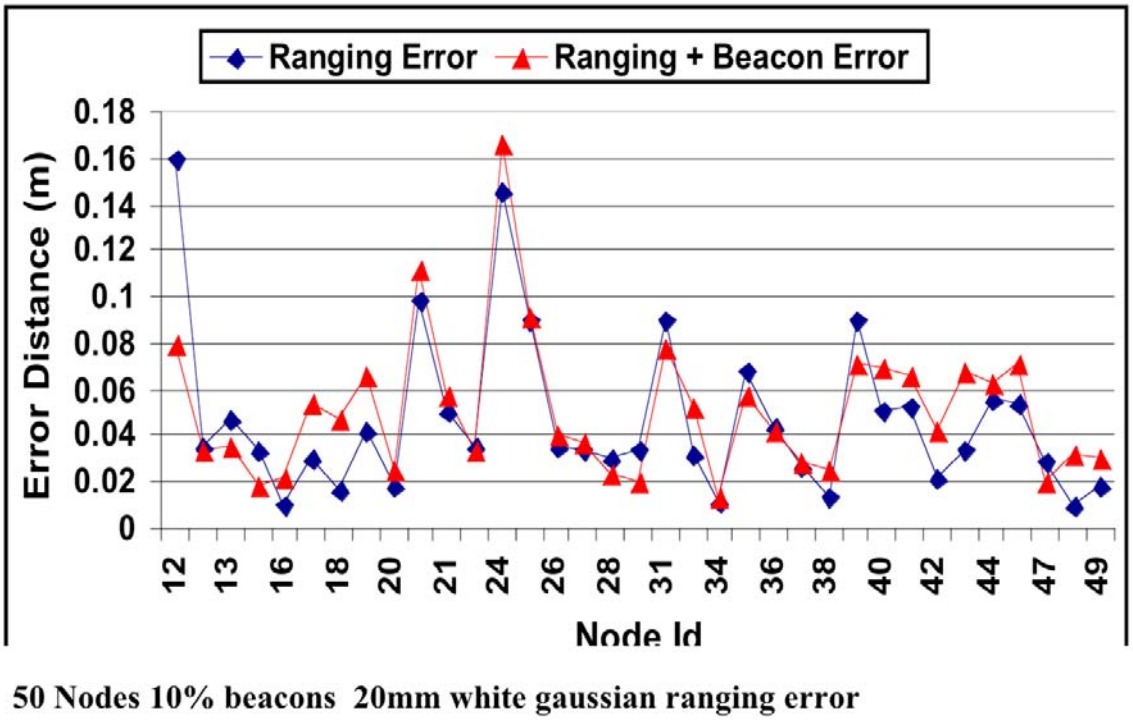


Figure 13: Iterative Multilateration Accuracy

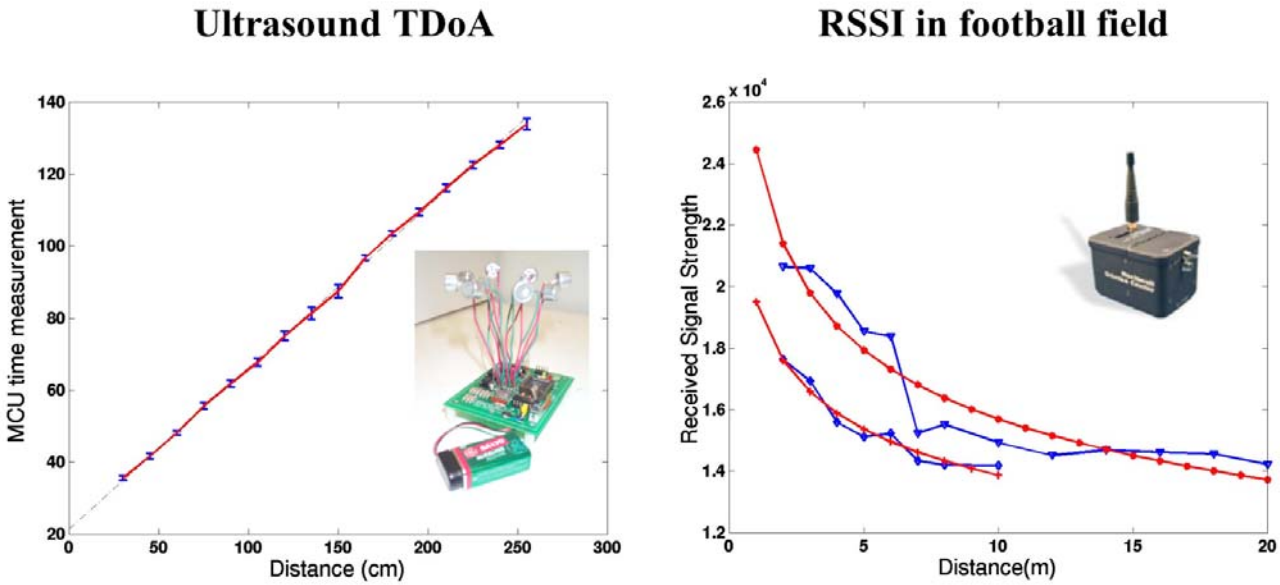


Figure 14: Ultrasound and RSSI Platform Characterizations

3 DECLARATIVE LANGUAGES AND EXECUTION ENVIRONMENT

3.1 Graphical Soldier Interfaces

Figure 15 provides an artists conception of a soldier graphical sensor network interface that supports simple tasking, status display, maintenance, and situation assessment functions. A number of technologies were converging that made a graphics-rich interfaces feasible without bulky computers. First, personal digital assistant devices were becoming widely available in the commercial marketplace and they have advanced to the point of having color displays, high performance 32-bit embedded processors, and megabytes of memory. Second, complete graphical libraries were migrating to these platforms, including Java and X-Windows. The PDA platforms are described further in Section 4. This convergence of technologies enabled the DSN team to explore mechanisms for having soldiers interact with the sensor network.

Figure 15: Artist Conception of Graphical User Interface

The DSN team collaborated with the BBN system integrator contractor to provide a graphical front-end which interfaced to query and tasking libraries being developed within the SensIT community. The distributed database query system being developed at the University of Maryland was the primary integration target in support of the SITEX SensIT program field experiments. The graphical interface was also was intended to support the SensorSim simulation tool to control and visualize simulated sensor fields. Other tools were incorporated as needed, such as the UCLA coverage server and Virginia Tech placement server.

3.1.1 First Generation Graphical User Interface

The first generation of the user interface, shown in Figure 16, was implemented in Java2 for a Linux or Windows laptop. Migration to Personal Java under Windows CE was an eventual target for this implementation. Personal Java is a subset of Java2, however, the subset excluded many of the most useful graphics libraries from the implementation. In the end, this made Personal Java not feasible as a target. Nevertheless, full Java implementations were promised for the IPAQ PDA either under WinCE or Linux. The graphical user interface was an extensible object-oriented Java architecture that supported a number of display and query mechanisms. For query, the user could highlight a region using a mouse or pen on the touch screen and tap for a pop-up menu of stored queries. This was the primary query mechanism. Queries could also be manually entered in a text-based dialog. The display system supported pan and zooming mechanisms and overlaid sensor field status information over a topographical vector map or registered satellite image. The tool can visualize sensor detection and target track events in real

time. For maintenance, the tool can display sensor node GPS-reported locations, id, battery status, and sensor configuration.

Figure 16: First Generation GUI

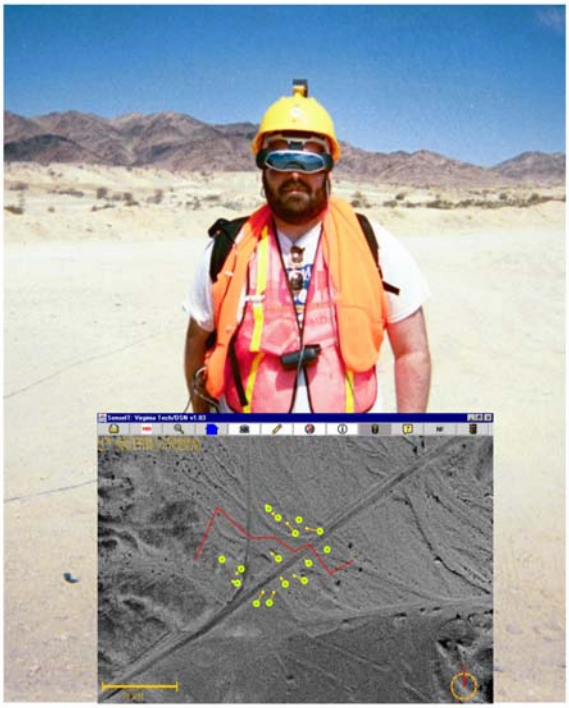


Figure 17: SITEX00 Experiment

This first generation GUI was used in support of the BBN-led SensIT SITEX00 experiment at 29 Palms, CA in August 2000. The GUI was integrated with the sensor field coverage algorithms and software developed by UCLA. At 29 Palms, the GUI communicated with the sensor field to acquire sensor node positions; these positions were fed to the sensor field coverage algorithms to provide calculation and visualization of sensor field coverage. Sensor nodes were moved to demonstrate that the software could respond to changes in the sensor field. As a second part of the experiment, the GUI was operated in conjunction with a digital compass, GPS, and head-mounted display, allowing the user to navigate the sensor field. The map rotated to maintain proper orientation for the operator. This combination of the sensor field coverage algorithms and the user interface allowed the operator to accurately place sensor nodes to their best effect. In this mode, a user specifies an area to be monitored by sensors. This monitored area is analyzed using sensor deployment algorithms developed at Virginia Tech and a sensor deployment plan is created. The user is shown in the GUI how to place the sensors according to the deployment plan. Guiding the user through sensor placement requires knowing where the user is physically located, where the user is headed, and where the user is looking. The small box containing a GPS unit, a digital compass, and other user-related sensors provided the user with this feedback. Using this interactive GLASTRON heads-up display, the researchers were able to navigate, view, and measure the SITEX00 sensor field. This version of the software ran on a laptop in a backpack. A GPS antenna was mounted on the backpack strap at the shoulder, and a digital compass was located on the helmet. A photo from the SITEX00 field experiment is given in Figure 17. This first generation GUI was used in the subsequent field exercises, but did not achieve the goal of running on a PDA.

Unfortunately, a robust Java distribution never materialized for PDAs and the performance of interpreted Java distributions never performed well enough to support the load of a graphical user interface. It had been hoped that the GUI would easily port to the iPAQ, but it did not. Java programs are intended to be portable, but they are only portable to platforms for which a Java virtual machine (Java VM) exists. The team experimented with the Kaffe (kaffe.org), a clean-room Java VM implementation, but it was not mature enough to support all Java features used by the GUI. There was a large effort at Compaq to port officially licensed Java VM, but an unfortunate clause in Sun's license agreement prevented the team from experimenting with these ports. The clause prohibits redistribution until a port passes 100% of a provided test suite. At the time, it passed all but a handful of these tests but was unavailable. Even when a Java VM did become available for the iPAQ, none of the initial versions had a sophisticated "just-in-time" JIT compiler so there were considerable performance problems. The DSN GUI ran so slowly it was unusable. It was not clear that the Java VM performance problems would be fixed soon, so an alternative approach was investigated.

3.1.2 Second Generation Graphical User Interface

The second generation DSN GUI was motivated by a desire to demonstrate a sensor network interface running on a handheld computer. It was also motivated by a desire to improve the map-handling capabilities of the GUI itself, allowing more types of data to be displayed and having a more structured interface with the sensor network. This was achieved by integrating the GUI with an open-source geographical information system called GRASS and an open-source SQL database server called PostgreSQL. When examining the requirements of the GUI, such as raster and vector import, display, map registration, and geographical browsing, it was realized that many of these features are standard in geographic information systems (GIS). In order to avoid duplicate work, it was decided to explore the possibility of adapting an existing GIS system to implement the GUI. GRASS (Geographic Resources Analysis Support System) is an open source, free software GIS system originally developed in the early 1980s by the U.S. Army Corps of Engineers' Construction Engineering Research Laboratory (USA/CERL). GRASS continues to be actively maintained today by a group of users/developers without involvement from USA/CERL. GRASS is available for Linux and uses the X Window System, although it had never been used on an iPAQ before the DSN team attempted to do so. Getting GRASS to work on the iPAQ was a relatively straightforward recompile for the ARM processor. It worked without modification. This is strong confirmation of the decision to use Linux on the iPAQ to improve portability.

GRASS provided the ability to import a tremendous amount of existing GIS data into the DSN GUI. GRASS supports a myriad of common, (and many not-so-common), GIS data formats so that it is a simple matter to import raster and vector layers from sources such as the USGS. For example, for SITEX00, the team had access to digital orthophoto quadrangle (aerial photography), man-made features, roads, vegetation, non-vegetative features, water, and topography. The second generation GUI operates by using native GRASS commands to display one or more of these static data layers. The GUI also uses the GRASS library functionality to draw dynamic sensor network data on top of the static layers. Dynamic data includes icons for sensor nodes and users and sensor network detection results. This second version of the GUI used a PostgreSQL database as the data interface from the sensor network to the GUI. This provided a standard, well-documented interface layer, which simplified the task of integrating sensor network and GUI software developed independently. The PostgreSQL database also

provided the GUI with a persistent record of all sensor network events for later replay. The PostgreSQL interface did not provide a mechanism for tasking the sensor network from the GUI.

This second generation interface was used at SITEX02 at 29 Palms in November 2001. It supported pan and zoom functionality of the original Java GUI and the ability to enable or disable various registered static and dynamic data layers. Using the extensive scripting language and graphical data manipulation utilities available in GRASS, the GUI also supported 3D perspective projections. A collection of screen shots is given in Figure 18. Although the system proved to be extremely flexible, it was also somewhat cumbersome. GRASS is intended for map manipulation and geographic data overlay for a wide variety of disciplines, such as geography, geology, oceanography, and climatology. It's support for dynamic data streams was extremely limited and the refresh rate of the map was one frame per second at best.

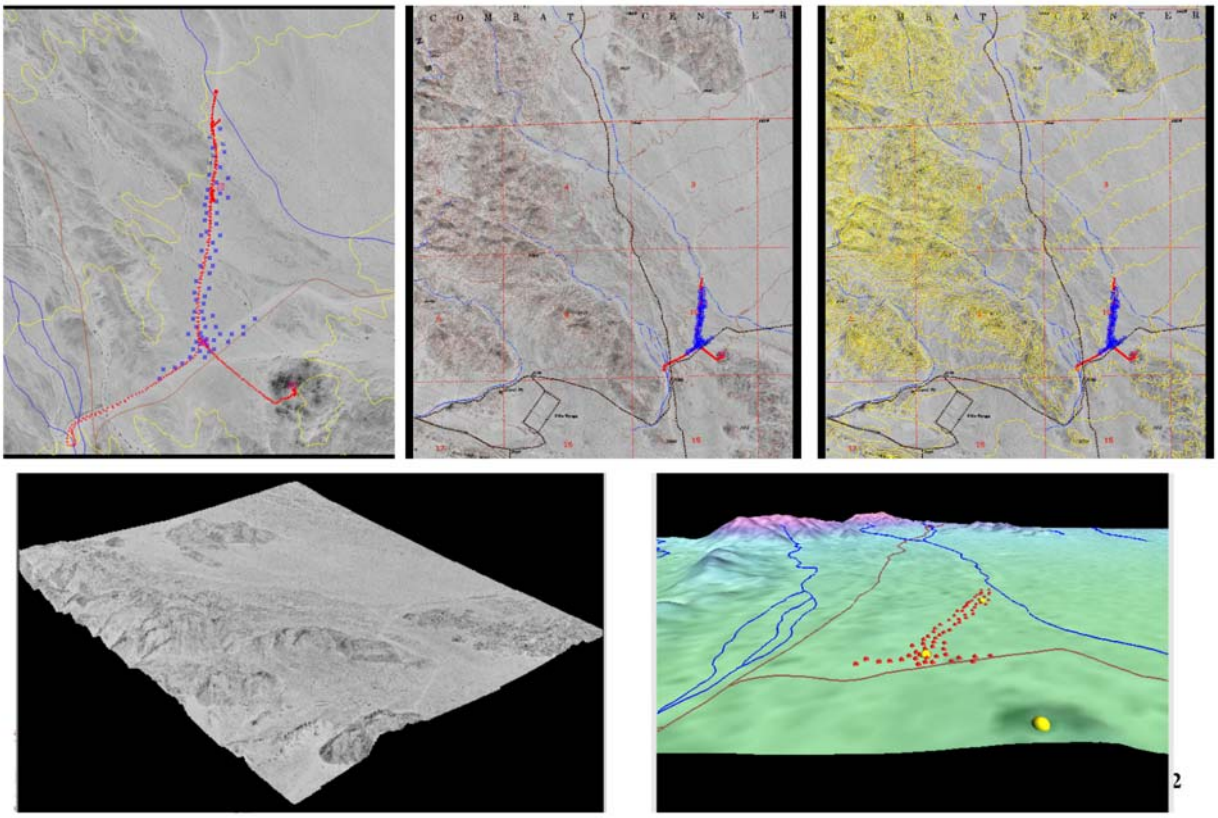


Figure 18: GRASS GUI

3.1.3 Third Generation Graphical User Interface

The third and final generation of the GUI, called GeoTV, builds on the capabilities of the previous generations while adding increased temporal browsing capabilities, better graphics support, and more general applicability. The fundamental motivation behind GeoTV is the recognition that many aspects of the sensor network visualization problem are dynamic in nature because of changes in network configurations, target activity and user queries. These dynamic qualities lend a temporal aspect for much of the data produced and consumed by the GUI. However, many GIS and RDBMS systems lack good support for dealing with temporal data. GeoTV provides the ability to browse geographic data, (as in a typical GIS system), but it also

allows the user to browse temporally. This "geo-temporal visualization" provides the user with a birds-eye map and a timeline. Each of these elements provides an overview of available global information in both space and time. The user indicates a region of interest on the map and a region of interest on the timeline. GeoTV responds by rendering a detailed view of the desired regions in the main panel.

The simple selection of regions on the map and timeline provide a powerful means for the user to graphically specify complex queries. For example, questions such as "Where have vehicles passed near this building in the past hour?" or "How many people have been detected in a given room over the past day?" can easily be answered with a few taps of a stylus on the map and timeline. Once GeoTV displays a focused region, the user can pan or zoom spatially and can also browse through time using VCR-like controls. When new data is actively being retrieved from a sensor network, this allow the user to request real-time visualization tasks such as "Display all currently detected vehicles including their tracks for the last 2 minutes."

Figure 19: GeoTV at BAE SYSTEMS

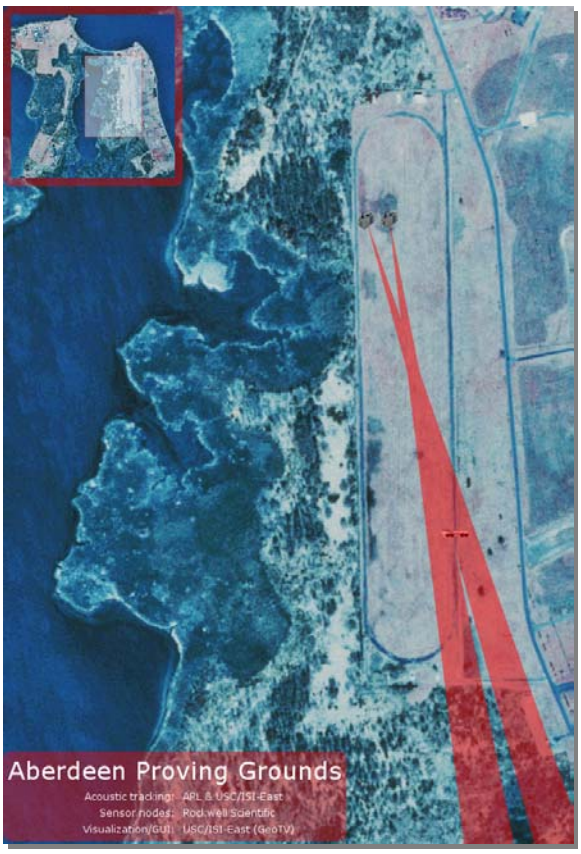


Figure 20: GeoTV at Spesuti Island

GeoTV has sophisticated rendering support. It can render SVG content for vector data layers as well as icons. All vector rendering is antialiased, which gives increased effective pixel resolution. This is important with displays such as that on the iPAQ that have many fewer pixels than a typical desktop display. GeoTV graphics can also include translucence, which has been used to good effect to indicate temporal separation of spatially coherent events. The graphics libraries developed for GeoTV are generally useful and are available to the open-source from http://www.xsvg.org. Unlike previous versions, GeoTV does not depend on the large external software packages required such as GRASS and PostgreSQL. GeoTV includes built-in scalable

graphics support. GRASS is used as an offline manner for importing GIS data since GeoTV accepts only a limited number of vector data formats (DXF and SVG) while GRASS can convert to DXF from many different formats. In the place of PostgreSQL interface, GeoTV interfaced directly to the COUGAR query interface system developed by Cornell University (http://cougar.cs.cornell.edu/). A preliminary version of GeoTV was demonstrated at the November 2002 PI meeting in Boston with data received over the Internet from BAE Systems in Austin, TX (shown in Figure 19). After the DSN project, GeoTV was used to display results from a DARPA PAC/C program experiment at Spesuti Island, Aberdeen Proving Ground, in Maryland. This display is shown in Figure 20. GeoTV continues to be actively developed and is available from http://www.geotv.org.

3.2 Sensor Placement Tool

Virginia Tech developed a sensor node placement optimization tool the addressed the problem of how to place and deploy sensors for a given set of application requirements in a specified geographic region. The application requirements include a number of diverse factors, such as target type, sensor type, terrain effects, errors in deployment, and coverage and redundancy goals. In terms of sensor types, sensors vary in range, accuracy, power, and detection probability for different kinds of targets. Targets under consideration range from personnel to vehicles. Likely target behavior can also be a factor (speed range, weight, loudness). Initial work on the coverage server was performed using the NS-2 sensor network simulator. An algorithm has been created that "seeds" the nodes onto the map; this seeding process is iterative, includes a random component, and is biased by the features on the map.

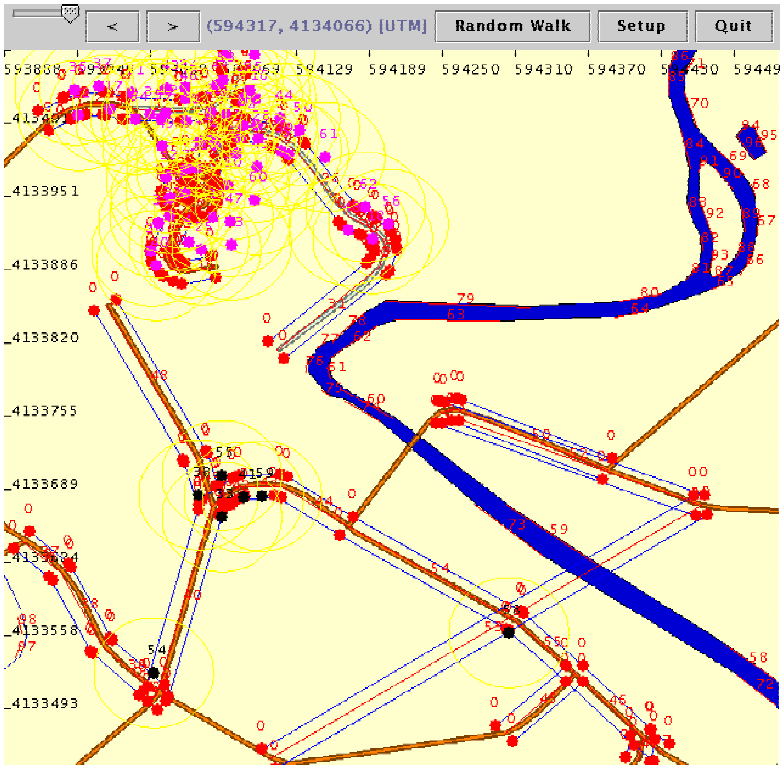


Figure 21:Terrain-Driven Placement Scenario

The placement tool was evaluated in a variety of scenarios.

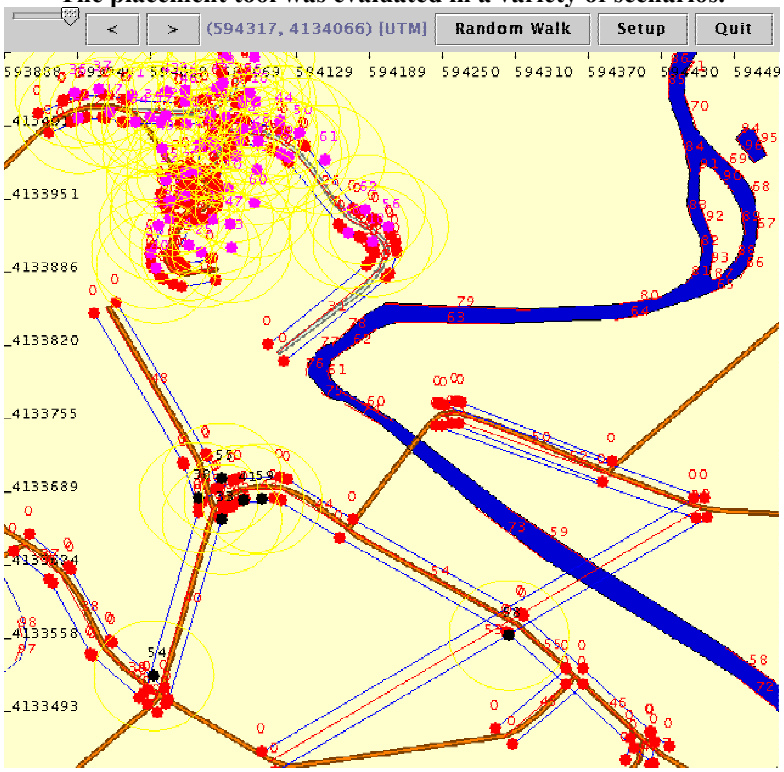


Figure 21 shows the visualization interface of the sensor placement tool for one such scenario. In this scenario, the placement engine was instructed to favor roads and avoid rivers. The performance on the tool was measured on several metrics, including the quality of coverage, the tolerance to faults of the system, and the connectivity of the resulting wireless network. Note that the tool does not directly optimize these metrics, but they are related to the optimizing function. In addition, the run time is linear in terms of the number of nodes and the number of geographic features. The tool takes Spatial Data Transfer Standard (SDTS) maps as input. Figure 22 shows an alternative scenario for redundant perimeter protection and Figure 23 shows a contour coverage computation view. Further details of the coverage server tool are provided in the addendum.

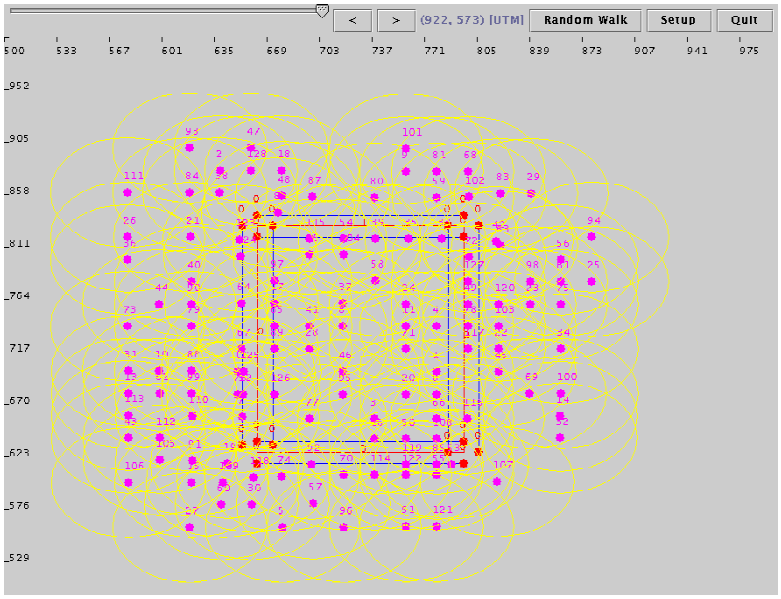


Figure 22: Redundant Perimeter Placement Scenario

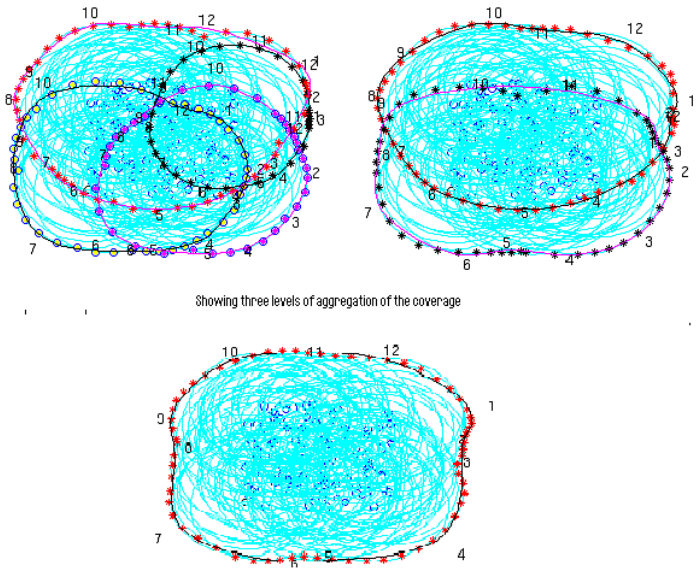


Figure 23: Coverage Contour Generation

4 PLATFORMS

The platform task in the DSN project investigated hardware prototypes and system software infrastructure for mobile soldier interfaces and prototype microsensors. The primary focus of this task centered on Linux development for embedded processor systems using the Compaq IPAQ™ personal digital assistant. This work is described in Section 4.1. The other major task included a communications of a GPS-synchronized radio, which is described in Section 4.2. The report concludes with a summary of DSN field experiments in Section 5.

4.1 Embedded Linux on IPAQ™

The Compaq IPAQ™ 3600 shown in Figure 24 is a commercial personal digital assistant (PDA) built on the StrongARM 32-bit embedded processor. This model has 32MB of SDRAM, 32MB of Flash, and a 320x200 color display. The IPAQ also has a connector on the back for a variety of expansion sleeves, such as Compact Flash and PC Card (PCMCIA). This PDA was unique at the time in that it supported the open-source Linux operating system. The DSN team realized that this would be an ideal prototyping platform for the soldier user interface. It has a processor with enough capability to run operating systems originally designed for larger desktop computers. The color display is important for communicating more information in fewer pixels. The reflective display makes outdoor viewing quite easy, even in direct sunlight, unlike traditional laptop displays that lose contrast in sunlight. Perhaps most importantly, the IPAQ is unique among handheld computers in that it provides a broad expansion connector, making much of the processor system bus available. This is very appealing as it could be used to develop a custom sleeves tailored for sensor network applications.

Figure 24: IPAQ 3600 Running Linux

4.1.1 DSN Linux Contributions

The open-source Familiar distribution at www.handhelds.org is maintained in part by researchers at Compaq Cambridge Research Laboratory. The DSN team added a number of tools to this distribution to make it robust for sensor network applications, including:

- ISI created a package management system called IPKG. This system is similar to Debian DPKG or RedHat RPMs, but tailored to handheld systems with no hard disk, just a flash file system. IPKG became the standard package management system for embedded Linux computers using the Familiar distribution and is used by a number of commercial embedded systems (http://www.handhelds.org/z/wiki/iPKG).
- ISI developed a full-screen pen-based stroke recognizer (XStroke) for text and command entry directly on the IPAQ display. It was developed for the GeoTV map display interface to input single-letter commands. This is now the default character input mechanism for the IPAQ Familiar Linux distribution (www.xstroke.org).
- ISI has contributed code to support trapezoid drawing in the Xfree86 Render Extension, later adapted into Cairo Vector Graphics Library (www.cairographics.org). For DSN, this work enabled anti-aliased display of translucent objects on the GeoTV GUI. For the wider open-source community, this contribution is beginning to have a significant impact on X-Window systems for most desktops by adding translucency to X applications, window managers, etc.

- ISI developed a Scalable Vector Graphics Library (XSVG) for rendering SVG vector images. For the GeoTV GUI, this library is used for vector data (roads, rivers, buildings, floorplans, etc.) and dynamic data such as detection and track events. This library is available at www.xsvg.org and is being actively used by application environments such as GNOME and KDE.

4.1.2 Video Surveillance using IPAQ

One of the applications of investigation under the DSN project was video surveillance and conferencing using portable handheld devices. The development platform consisted of a Compaq IPAQ 3600, a dual PC card sleeve, a Cabletron 802.11b PC card, and a Videum PC Card camera. The camera supported 320x200 color images at up to 15 frames per second. A photo of the platform is shown in Figure 25 (left). A power consumption breakdown is given in Figure 25 (right). Transmitting 15 frames per second, the entire system consumed less than 1.2 Watts including the processor (172mW), camera (533mW), and wireless card (357mW). This entirely COTS prototype compared well with research microsensor platforms used solely for acoustic data processing and indicated that video is quite possible with the state of the art.

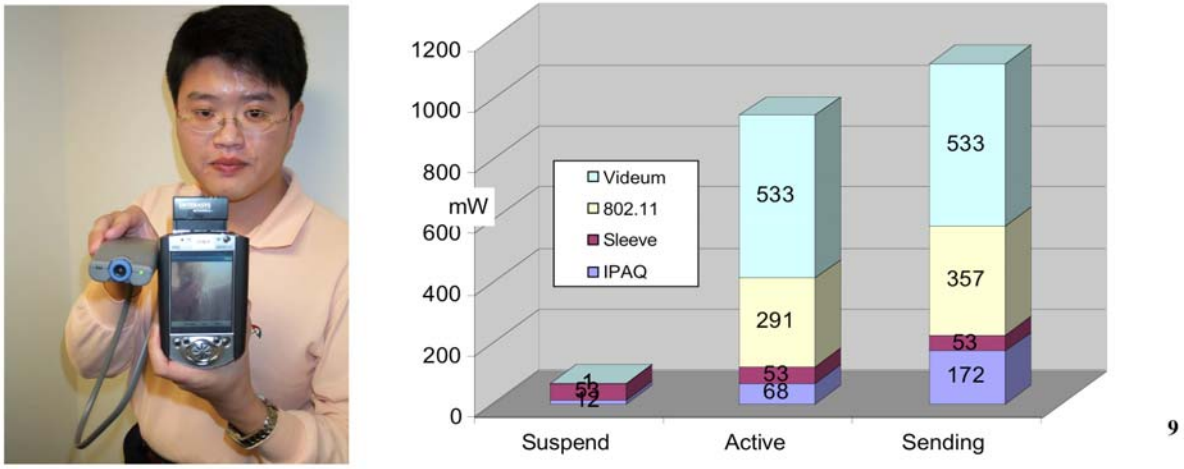


Figure 25: DSN Video Platform

The software infrastructure used for video surveillance experiments was based on open-source video conferencing software package called *vic*. This software was ported from desktop Linux to the IPAQ. A number of optimizations were introduced to make this port feasible. Since Linux usually runs on desktop processors with IEEE floating point, many of the video coder/decoders (CODECS) did not perform well without floating point on the IPAQ. Table 2 gives the achievable frame rates using an IPAQ for different scenes. The best CODEC, *nv,* was selected for the SensIT field experiments.

Table 2: Vic CODEC Frame Rate Analysis

Codec	Lens covered		Still scene		Moving camera		Image Quality (subjective)
	Frame rate	Network traffic	Frame rate	Network traffic	Frame rate	Network traffic	
nv	18 fps	90 kb/s	18 fps	100 kb/s	18 fps	1200 kb/s	Good
nvdct	18 fps	91 kb/s	18 fps	67 kb/s	18 fps	680 kb/s	Good
cellb	18 fps	28 kb/s	18 fps	30 kb/s	18 fps	800 kb/s	Poor
jpeg	0.5 fps	5 kb/s	0.5 fps	8 kb/s	0.5 fps	8 kb/s	Fair
h261	17 fps	46 kb/s	17 fps	30 kb/s	0.7 fps	9 kb/s	Good
bvc	18 fps	19 kb/s	18 fps	18 kb/s	18 fps	500 kb/s	Good
h263+	0.1 fps	1 kb/s	0.1 fps	1 kb/s	0.1 fps	1 kb/s	Fair
h263	15 fps	18 kb/s	15 fps	14 kb/s	1.3 fps	25 kb/s	Fair

4.2 GPS-Synchronized Communications

ISI constructed an experimental platform to evaluate the idea of using GPS to synchronize radios in a sensor network. Some GPS units provide a 50ns-accurate clock reference and the concept was to use this timing pulse to enable fine-grained TDMA communications. The DSNCOMM board block diagram is given in Figure 26.

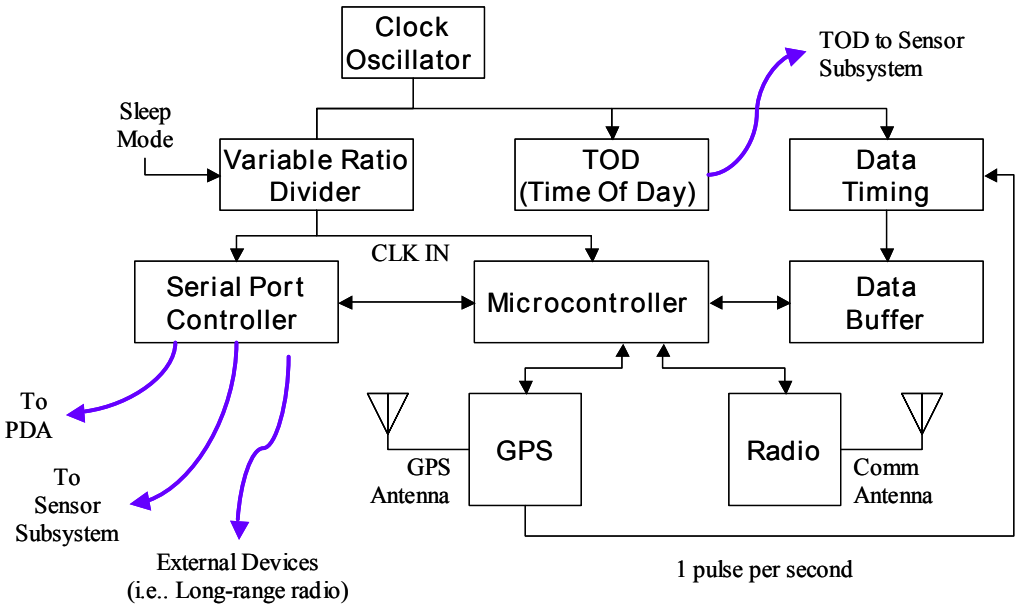


Figure 26: DSNCOMM Block Diagram

A photo of the DSNCOMM board for GPS-synchronized communications is given in Figure 27. The board uses a commercial Motorola Oncore UT GPS daughter board that can provide the 50-ns accurate 1 pulse-per-second clock reference. A firmware update of the GPS chipset allows 100 pulses-per-second, which enables finer granularity TDMA slots. The radio is a 100 kbps 900 MHz band RFMD 9901/9902 FSK radio. Five DSNCOMM boards were fabricated under the DSN project. Figure 28 is a screen capture of the node transmitter output spectrum, which is generating a 60 kHz square wave through the radio.

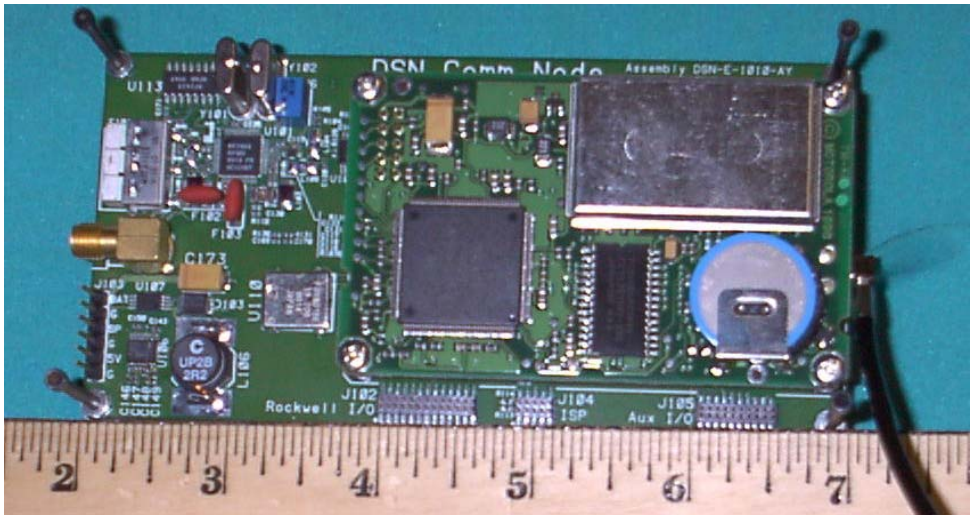


Figure 27: DSNCOMM Board

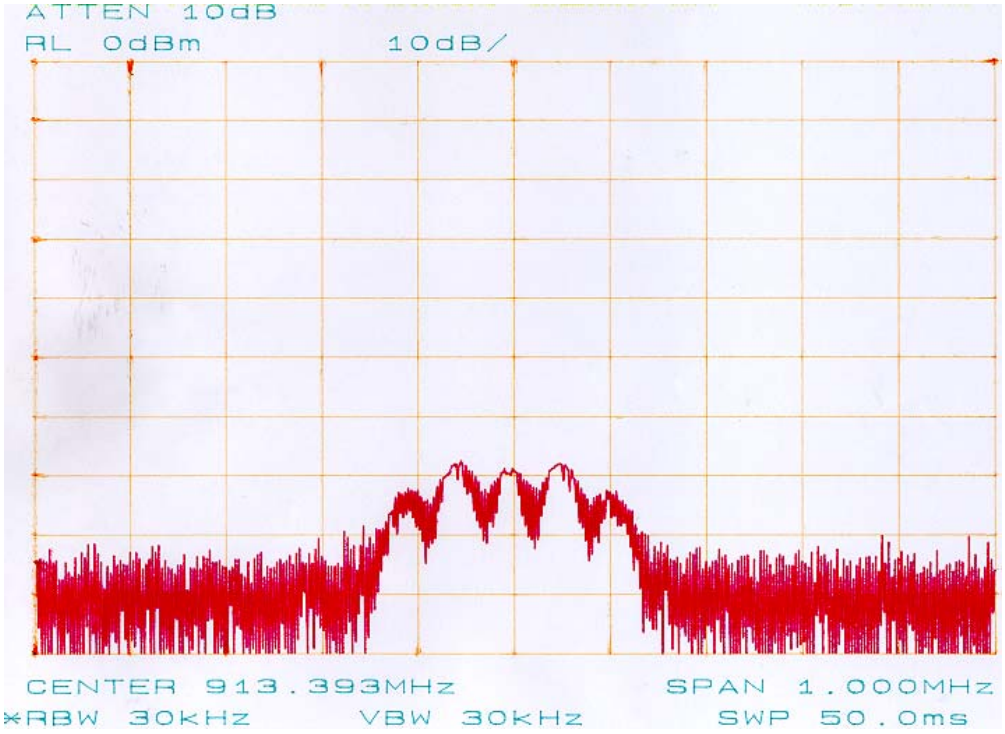


Figure 28: DSNCOMM Radio XMIT Spectrum

The concept of operations is to allow the sensor nodes to remain off for extended periods of time and support very narrow time windows where communications exchanges occur. GPS is the only mechanism that doesn't drift because of thermal or other environmental conditions, so time synchronization is possible to a very fine degree. The operational states of the radio are given in Table 3. The energy measurements for the various states are provided in Table 4. The total energy consumed per packet is 37.2165 Joules. For a packet size of 10,000 bits (100ms @ 100kbps), the energy per bit is 3.7E-3 joules/bit.

Table 3: DSNCOMM Communication States

SYNC	GPS does cold start; acquire satellite almanac, ephemeris, and time of day; GPS turned off.
SLEEP	Set the timer for 30 seconds before scheduled transmit time, then sleep the processor.
WAKE	When timer trips, power up the processor and then turn on the GPS.
SETTLE	Five milliseconds before transmission turn on the transmitter and allow the PLL to settle.
TRANSMIT	At the 1 PPS edge, transmit for 100 milliseconds at 100 kbps. Total data transfer is 10000 bits.
SLEEP	Turn off the transmitter and sleep the processor.

Table 4: DSNCOMM Energy Estimates

Item	Duration	Voltage	Current	Watts	Joules
Initialization (done once per day or less)					
GPS Cold Start	300 sec	5V	180 mA	.9 W	270
Operation (per packet)					
Processor sleep (with timer running)	4 hours	5V	100e-6 A	0.5 mW	7.2
Processor on and GPS reacquire	30 sec	5V	200 mA	1 W	30
TX Settle	10 ms	5V	30 mA	0.15 W	0.0015
TX Data	100 ms	5V	30 mA	0.15 W	0.0150
Total Joules per packet					37.2165

5 FIELD EXPERIMENT SUMMARY

DSN participated in all three SensIT field experiments and other demonstrations

5.1 SITEX00, 29 Palms, August 2000

The DSN team participated in SITEX00 in August 2000 at Twenty-Nine Palms, CA. A prototype user platform was fielded using a laptop, heads-up display, GPS, and digital compass. The first generation GUI (see Section 3.1.1) was used to navigate the sensor field and visualize changes to the maximal breach path in real time.

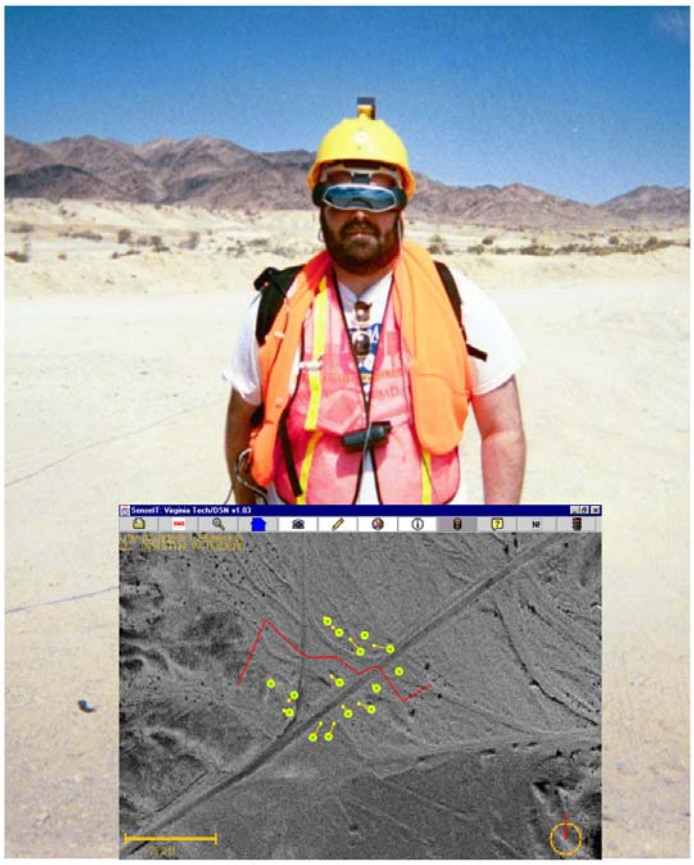


Figure 29: SITEX00 Experiment

5.2 SITEX01, 29 Palms, March 2001

At SITEX01 in March 2001, the DSN team collaborated with the Rockwell-led SenosrWare project. The experiment consisted of ten Rockwell HYDRA platforms with geophones and one laptop with a webcam video camera. The HYDRA nodes performed a wave intensity comparison algorithm (WIC) to generate bearings to the target. The results were displayed on the first generation GUI. The experiment also included a video surveillance demonstration. The laptop generated the camera data, but the results were displayed on a IPAQ for the first time. The DSN team also supported the main BBN/Sensoria experiment using the first generation GUI. A summary of the experiment is shown in Figure 30.

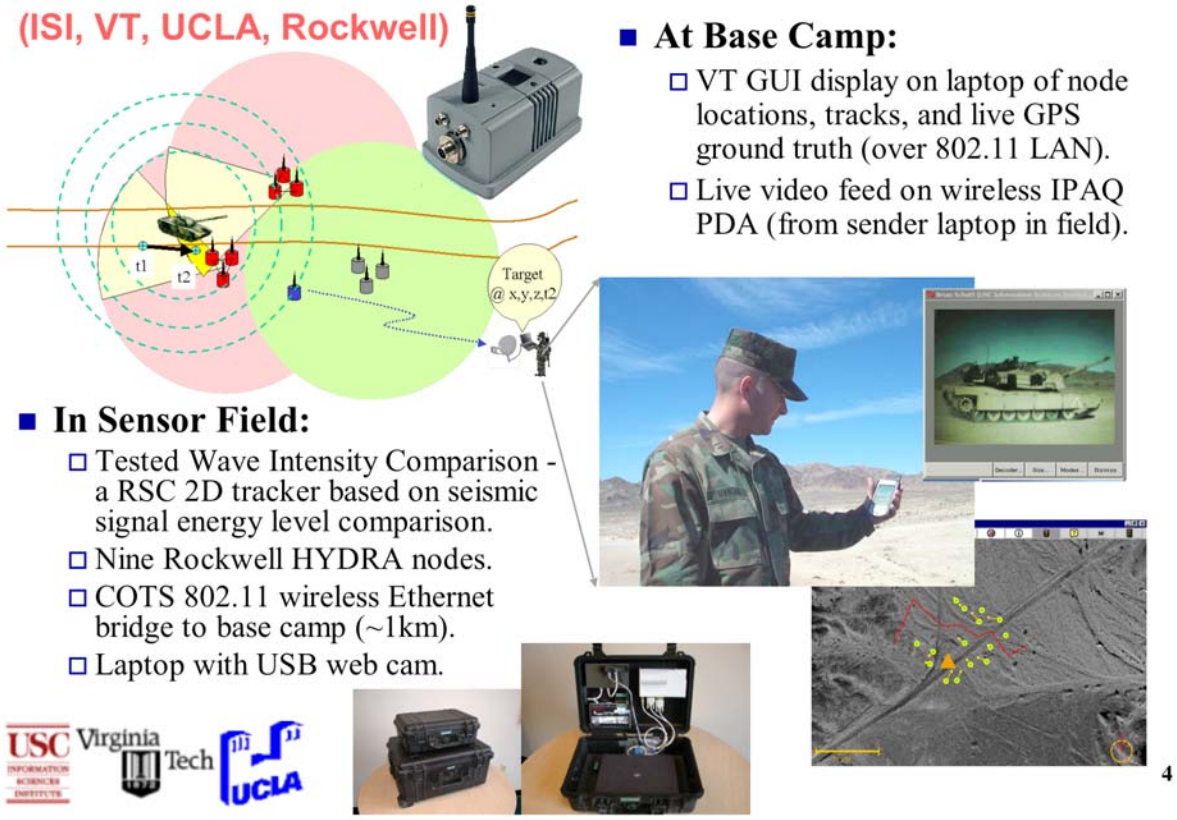


Figure 30: SITEX01 Experiment

5.3 SITEX02, 29 Palms, November 2001

At SITEX02 in November 2001, the DSN team fielded three prototype IPAQ platforms and collected video data for tracked and wheeled vehicles. Live GPS from DSN-instrumented vehicles was displayed on the second generation GUI. DSN also supported the main BBN-led experiment with Sensoria 2.0 nodes using the first generation Java GUI. A number of secondary experiments for coverage algorithms were also performed. A summary of the experiment is provided in Figure 31 and example video is shown in Figure 32.

Figure 31: SITEX02 Experiment

Figure 32: SITEX02 Video Frames

6 DELIVERABLES SUMMARY

6.1 Deliverables for FY99

6.1.1 Distribution and Aggregation

1) Initial Network Services API Specification (UCLA) [**complete**]
UCLA has defined a set of functions that make up this API specification for the DSN platform and continues to analyze the underlying protocols using the *ns* simulation tool.

6.1.2 Declarative Languages and Execution Environment

2) Topographical Map GUI Prototype Specification (VT) [**complete**]
Virginia Tech delivered a first release of the GUI that is able to process user inputs and generates a format appropriate for U-Maryland query language.

3) Query Language Integration Specification (VT) [**complete**]
Virginia Tech has been working with the SenseIT community at the BBN telecons and has specified an interface to generate user inputs for the U-Maryland query language.

6.1.3 Platforms

4) DSN Research Platform Specification (USC/ISI) [**complete**]
USC/ISI has completed the design of the comm. subsystem board.

6.2 Deliverables for FY00

6.2.1 Distribution and Aggregation

1) NS Simulation Code Release and Documentation (UCLA) [**complete**].
UCLA has made the SensorSim simulator code available to other members of the SensIT community. USC/ISI (Deborah Estrin's group) is making SensorSim a formal part of the ns release.

2) Spatial Addressing and Routing Simulation (UCLA) [**complete**].
UCLA is investigating addressing and routing protocols currently using the *ns* simulator. This work is exercising the *ns* simulator development. The simulation work is complete; an implementation using a test platform is also being done.

6.2.2 Declarative Languages and Execution Environment

3) Java Code Release and Documentation (VT) [**complete**].
Virginia Tech has delivered the GUI source code to BBN. VT demonstrated this code at the SITEX00 experiment. Periodic updates continued for SITEX01 and SITEX02 experiments.

6.3 Deliverables for FY01

6.3.1 Distribution and Aggregation

4) PDA Experiment Code Release, Documentation, Report (UCLA) [**complete**].
UCLA has completed an implementation of sensor data distribution protocols on a network of iPAQ PDAs using 802.11b radios, and serial magnetometers. Another implementation using prototype radios based on RFM transceivers was done.

6.3.2 Declarative Languages and Execution Environment

Integration Code Release, Documentation, Report (VT) [**Complete**].
Virginia Tech released new versions of the GUI to BBN's specifications for use in the SITEX02 experiment at Twentynine Palms as well as for the 2002 demonstration at the PI meeting. This has been tested with the UMD gateway simulator and integration with other pieces of the processing chain continues.

6.3.3 Platforms

5) Integrated Platform Selection Report (ISI) [**Complete**].
The SensIT community has specified the Sensoria 2.0 platform. The DSN team is using this platform for experiments where appropriate and relying on IPAQ PDAs for secondary experiments.

6) GPS Experiment Report (ISI) [**Complete**].
Results are included in the DSN final report.

6.4 Deliverables for FY02

6.4.1 Distribution and Aggregation

1) Integration Code Release, Documentation, and Report (UCLA) [Complete].
UCLA has released a snapshot of the mobile sensor script framework, operational on iPaqs and Sensoria WINS nodes, via SourceForge.

6.4.2 Platforms

2) Integrated GPS experiment (ISI) [Complete].
The final integrated experiment was performed in conjunction with Cornell and BAE SYSTEMS (Austin) in November 2002.

7 PERSONNEL

7.1 USC Information Sciences Institute Personnel

- Brian Schott — Project Leader, PI
- Robert Parker — Director, ISI-E
- Joe Czarnaski — Researcher
- Doe-Wan Kim — Researcher
- Bruce Parham — Engineer
- Ron Riley — Researcher
- Jack Wills — Researcher
- Carl Worth — Researcher

7.2 UCLA Personnel

- Mani Srivastava — Associate Professor, Co-I
- Athanassios Boulis — PhD Student
- Gautam Kulkarni — PhD Student
- Sung Park — Graduate Student
- Andreas Savvides — PhD Student
- Curt Schurgers — PhD Student
- Vlassis Tsiatsis — PhD Student
- Scott Zimbeck — Graduate Student

7.3 Virginia Tech Personnel

- Mark Jones — Associate Professor, Co-I
- Peter Athanas — Assistant Professor, Co-I
- Arya Abraham — Graduate Research Assistant
- Gary Friedman — Undergraduate Research Assistant
- Dennis Goetz — Undergraduate Research Assistant
- Anup Gupta — Graduate Research Assistant
- Christian Laughlin — Undergraduate Research Assistant
- Shashank Mehrotra — Graduate Research Assistant
- Jonathan Scott — Undergraduate Research Assistant
- Manu Sporny — Undergraduate Research Assistant

8 PUBLICATIONS

[1] "Dynamic Fine-grained Localization in Ad-hoc Networks of Sensors" by Andreas Savvides, Chih-Chieh Han, and Mani Srivastava. UCLA Technical Report, and paper submission to Mobicom 2001.

[2] "On modeling networks of wireless microsensors" by Andreas Savvides, Sung Park, and Mani Srivastava. UCLA Technical Report, and paper submitted to Sigmetrics 2001.

[3] "Tasking Distributed Sensor Networks" Mark Jones, Shashank Mehrotra, and Jae Hong Park to Journal of High Performance Computing. Accepted for publication.

[4] S. Park, A. Savvides, and M. Srivastava, "Simulating networks of wireless sensors," Proceedings of the 2001 Winter Simulation Conference (WSC 2001), December 2001.

[5] C. Schurgers, and M. Srivastava, "Energy efficient routing in wireless sensor networks," Proceedings of MILCOM 2001, October 2001.

[6] C. Schurgers, G. Kulkarni, and M. Srivastava, "Distributed assignment of encoded MAC addresses in wireless sensor networks," Proceedings of the ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2001), October 2001.

[7] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-aware wireless sensor networks", IEEE Signal Processing (special issue on collaborative signal processing), March 2002.

[8] C. Schurgers, V. Tsiatsis, and M.B. Srivastava, "STEM: Topology management for energy efficient sensor networks," IEEE Aerospace Conference, March 2002.

[9] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M.B. Srivastava, "Optimizing Sensor Networks in the Energy-Density-Latency Design Space," IEEE Transactions on Mobile Computing, vol. 1, (no. 1), January-March 2002. p. 70-80. 11 pages. [NOTE: This inaugural issue appeared late, in June 2002]

[10] C. Schurgers, G. Kulkarni, and M.B. Srivasatava, "Topology Management for Sensor Networks: Exploiting Latency and Density," The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2002), June 2002. 13 pages.

[11] S. Slijepcevic, V. Tsiatsis, S. Zimbeck, M. Potkonjak, and M.B. Srivastava, "On Communication Security in Wireless Ad-Hoc Sensor Networks," The IEEE Eleventh International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2002): Enterprise Security, June 2002. 6 pages.

[12] G. Kulkarni, C. Schurgers, and M.B. Srivastava, "Dynamic Link Labels for Energy Efficent MAC Headers in Wireless Sensor Networks," Proceedings of the First IEEE International Conference on Sensors, June 2002. 6 pages.

[13] Boulis, and M.B. Srivastava, "A Framework for Efficient and Programmable Sensor Networks," Proceedings of the Fifth IEEE Conference on Open Architectures and Network Programming (OPENARCH'02), June 2002. 12 pages

[14] A. Savvides, H. Park and M. B. Srivastava, " The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems", Proceedings of the First

ACM International Workshop on Sensor Networks and Applications held in conjunction with Mobicom, September 2002.

[15] A. Savvides and M. B. Srivastava, " A Distributed Computation Platform for Wireless Embedded Sensing", Proceedings of ICCD 2002, Freiburg, Germany, September 2002.

[16] Athanassios Boulis and Mani Srivastava, “Node-level Energy Management for Sensor Networks in the Presence of Multiple Applications”, IEEE International Conference on Pervasive Computing and Communications (PerCom), March 2003. (Accepted)

[17] Athanassios Boulis, Chih-Chieh Han, and Mani Srivastava, “Design and Implementation of a Framework for Efficient and Programmable Sensor Networks,” ACM MobiSys, 2003. (Accepted)

[18] Carl D. Worth. xstroke: Full-screen Gesture Recognition for X. In FREENIX Track: 2003 Annual Technical Conference, pages 187-196. June 2003.

[19] Carl D. Worth and Keith Packard. Xr: Cross- device Rendering for Vector Graphics. Ottowa Liniux Symposium. July 2003.

9 LIST OF ACRONYMS

- ARL – Army Research Labs
- ASK – Amplitude Shift Keying
- CODEC – Encoder / Decoder
- CPU – Central Processing Unit
- DARPA – Defense Advanced Research Projects Agency
- DSP – Digital Signal Processor
- FEC – Forward Error Correction
- FFT – Fast Fourier Transform
- FPGA – Field Programmable Gate Array
- HAL – Hardware Abstraction Layer
- ISI – Information Sciences Institute
- LOB – Line of Bearing
- MAC – Media Access Control
- MIT – Massachusetts Institute of Technology
- OOK – On/Off Keying
- OS – Operating System
- P-A – Power Aware
- PAC/C – Power Aware Computing and Communications
- PCMCIA – Personal Computer Memory Card International Association
- PSK – Phase Shift Keying
- QAM – Quadrature Amplitude Modulation
- RSC – Rockwell Scientific Company (formerly Rockwell Science Center)
- RTOS – Real Time Operating System
- STEM – Sparse Topology and Energy Management
- TDMA – Time Domain Multiple Access
- TI – Texas Instruments
- UCLA – University of California, Los Angeles
- USC – University of Southern California
- VLSI – Very Large Scale Integrated

10 LIST OF ADDENDA

[1] "Dynamic Fine-grained Localization in Ad-hoc Networks of Sensors" by Andreas Savvides, Chih-Chieh Han, and Mani Srivastava. UCLA Technical Report, and paper submission to Mobicom 2001.

[2] "On modeling networks of wireless microsensors" by Andreas Savvides, Sung Park, and Mani Srivastava. UCLA Technical Report, and paper submitted to Sigmetrics 2001.

[3] "Tasking Distributed Sensor Networks" Mark Jones, Shashank Mehrotra, and Jae Hong Park to Journal of High Performance Computing. Accepted for publication.

[4] S. Park, A. Savvides, and M. Srivastava, "Simulating networks of wireless sensors," Proceedings of the 2001 Winter Simulation Conference (WSC 2001), December 2001.

[5] C. Schurgers, and M. Srivastava, "Energy efficient routing in wireless sensor networks," Proceedings of MILCOM 2001, October 2001.

[6] C. Schurgers, G. Kulkarni, and M. Srivastava, "Distributed assignment of encoded MAC addresses in wireless sensor networks," Proceedings of the ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2001), October 2001.

[7] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-aware wireless sensor networks", IEEE Signal Processing (special issue on collaborative signal processing), March 2002.

[8] C. Schurgers, V. Tsiatsis, and M.B. Srivastava, "STEM: Topology management for energy efficient sensor networks," IEEE Aerospace Conference, March 2002.

[9] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M.B. Srivastava, "Optimizing Sensor Networks in the Energy-Density-Latency Design Space," IEEE Transactions on Mobile Computing, vol. 1, (no. 1), January-March 2002. p. 70-80. 11 pages. [NOTE: This inaugural issue appeared late, in June 2002]

[10] C. Schurgers, G. Kulkarni, and M.B. Srivasatava, "Topology Management for Sensor Networks: Exploiting Latency and Density," The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2002), June 2002. 13 pages.

[11] S. Slijepcevic, V. Tsiatsis, S. Zimbeck, M. Potkonjak, and M.B. Srivastava, "On Communication Security in Wireless Ad-Hoc Sensor Networks," The IEEE Eleventh International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-2002): Enterprise Security, June 2002. 6 pages.

[12] G. Kulkarni, C. Schurgers, and M.B. Srivastava, "Dynamic Link Labels for Energy Efficent MAC Headers in Wireless Sensor Networks," Proceedings of the First IEEE International Conference on Sensors, June 2002. 6 pages.

[13] A. Boulis, and M.B. Srivastava, "A Framework for Efficient and Programmable Sensor Networks," Proceedings of the Fifth IEEE Conference on Open Architectures and Network Programming (OPENARCH'02), June 2002. 12 pages

[14] A. Savvides, H. Park and M. B. Srivastava, " The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems", Proceedings of the First

ACM International Workshop on Sensor Networks and Applications held in conjunction with Mobicom, September 2002.

[15] A. Savvides and M. B. Srivastava, " A Distributed Computation Platform for Wireless Embedded Sensing", Proceedings of ICCD 2002, Freiburg, Germany, September 2002.

[16] Athanassios Boulis, Chih-Chieh Han, and Mani Srivastava, “Design and Implementation of a Framework for Efficient and Programmable Sensor Networks,” ACM MobiSys, 2003. (Accepted)

Appendix A:

Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors

Andreas Savvides, Chih-Chieh Han and Mani B. Strivastava
Networked and Embedded Systems Lab
Department of Electrical Engineering
University of Calfornia, Los Angeles
{asavvide, simonhan, mbs}@ee.ucla.edu

ABSTRACT

The recent advances in radio and embeddedsystem technologies have enabled the proliferation of wireless microsensor networks.Such wirelessly connected sensors are released in many diverse environments to perform various monitoring tasks. In many such tasks, location awareness is inherently one of the most essential system parameters. It is not only needed to report the origins of events, but also to assist group querying of sensors, routing, and to answer questions on the network coverage.In this paper we present a novel approach to the localization of sensors in an ad-hoc network.We describe a system called AHLoS (Ad-Hoc Localization System) that enables sensor nodes to discover their locations using a set distributed iterative algorithms. The operation of AHLoS is demonstrated with an accuracy of a few centimeters using our prototype testbed while scalability and performance are studied through simulation.

Keywords

location discovery, localization, wireless sensor networks

1. INTRODUCTION

1.1 Sensor Networks and Location Discovery

Nowadays, wireless devices enjoywidespread use in numerous diverse applications including that of sensor networks. The exciting new field of *wireless sensor networks* breaks away from the traditional end-to-end communication of voice and data systems, and introduces a new form of distributed information exchange. Myriads of tiny embedded devices, equipped with sensing capabilities, are deployed in the environment and organize themselves in an ad-hoc network. Information exchange among collaborating sensors becomes the dominant form of communication, and the network essentially behavesas a large, distributed computation machine. Applications featuring such networkeddevices are becoming increasingly prevalent, ranging from environmental and natural habitat monitoring, to home networking, medical applications and smart battlefields. Networked sensors can signal a machine malfunction to the control center in a factory, or alternatively warn about smoke on a remote forest hill indicating that a dangerous fire is about to start. Other wireless sensor nodes can be designed to detect the ground vibrations generated by the silent footsteps of a cat burglar and trigger an alarm.

Naturally, the question that immediately follows the actual detection of events, is: *where?* Where are the abnormal vibrations detected, where is the fire, which house is about to be robbed? To answer this question, a sensor node needs to possess knowledge of its physical location in space. Furthermore, in large scale ad-hoc networks, knowledge of node location can assist in routing [5] [6], it can be used to query nodes over a specic geographicalarea or it can be used to study the coverage properties of a sensor network [31]. Additionally, we envision that location awareness developed here will enjoy a wide spectrum of applications. In tactical environments, it can be used to track the movements of targets. In a smart kindergarten [32] it can be used to monitor the progress of children by tracking their interaction with toys and with each other; in hospitals it can keep track of equipment, patients,doctors and nurses or it can drive context aware services similar to the ones described in [4], [29].

The incorporation of location awarenessin wireless sensor networks is far from a trivial task. Since the network can consist of a large number of nodes that are deployed in an ad-hoc fashion, the exact node locations are not known apriori. Unfortunately, the straighforward solution of adding GPS to all the nodes in the network is not practical since:

- GPS cannot work indoors or in the presence of dense vegetation, foliage or other obstacles that block the line-of-sight from the GPS satellites.
- The power consumption of GPS willreduce the battery life on the sensor nodes thus reducing the effective lifetime of the entire network.
- The production cost factor of GPS can become an issue when large numbers of nodes are to be produced.
- The size of GPS and its antenna increases the sensor node form factor. Sensor nodes are required to be small and inobstrusive.

ACM SIGMOBILE 7/01 Rome, Italy

To this end, we seek an alternative solution to GPS that is low cost, rapidly deployable and can operate in many diverse environments without requiring extensive infrastructure support.

Figure 1: WINS Sensor Node from RSC

1.2 Our Work

We propose a new distributed technique that only requires a limited fraction of the nodes (beacons) to know their exact location (either through GPS or manual configuration) during deployment and that nevertheless can attain network-wide fine-grain location awareness. Our technique, which we call AHLoS (Ad-Hoc Localization System), relieves the drawbacks of GPS as it is low cost, it can operate indoors and does not require expensive infrastructure or pre-planning. AHLoS enables nodes to dynamically discover their own location through a two-phase process, ranging and estimation. During the ranging phase, each node estimates its distance from its neighbors. In the estimation phase, nodes with unknown locations use the ranging information and known beacon node locations in their neighborhood to estimate their positions. Once a node estimates its position it becomes a beacon and can assist other nodes in estimating their positions by propagating its own location estimate through the network. This process iterates to estimate the locations of as many nodes as possible.

The first part of our work examines the ranging challenges. Since almost all ranging techniques rely on signal propagation characteristics, they are susceptible to external biases such as interference, shadowing and multipath effects, as well as environmental variations such as changes in temperature and humidity. These physical effects are difficult to predict and depend greatly on the actual environment in which the system is operated. It is therefore critical to characterize the behavior of different ranging alternatives experimentally in order to determine their usefulness in sensor networks. To justify our rangining choice we performed a detailed comparison of two promising ranging techniques: one based on received RF signal strength and the other based on the Time of Arrival (ToA) of RF and ultrasonic signals. Our experiments of distance discovery with RF signal strength were conducted on the WINS wireless sensor nodes [12] (figure 1) developed by the Rockwell Science Center (RSC). To perform our evaluation of ToA, we have designed and implemented a testbed of ultrasound-equipped sensor nodes, called *Medusa* (from Greek mythology - a monster with many heads) nodes (figure 2). To address the variation of propagation characteristics of ultrasound from place to place AHLoS estimates the propagation characteristics on the fly in the actual deployment environment. The second part of our work uses the ranging techniques described above, to develop a set of distributed localization algorithms. Node positions are estimated using least squares estimation in an iterative multilateration process. This ability of AHLoS to estimate node locations in an ad-hoc setting with a few centimeters accuracy is demonstrated on a testbed comprised of first generation *Medusa* nodes. Error propagation, system scalability and energy consumption are studied through simulation.

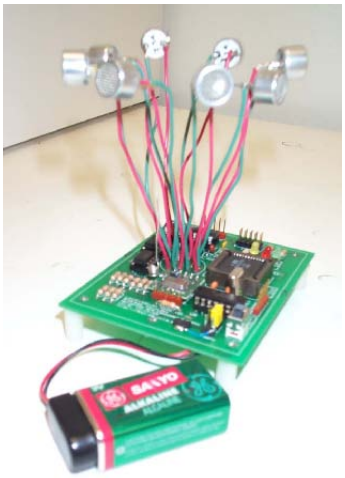


Figure 2: *Medusa* experimental node

1.3 Paper Organization

This paper is organized as follows: In the next section we provide some background on localization and we survey the related work. Section 3 presents the evaluation of our two candidate ranging methods: Received signal strength and time of arrival. Section 4 describes the localization algorithms and section 5 is a short study on node and beacon node placement. In section 6 we discuss our implementation and experiments. Section 7 discusses the tradeoffs between centralized and distributed localization and section 8 concludes this paper.

2. BACKGROUND AND RELATED WORK

2.1 Background

The majority of existing location discovery approaches consist of two basic phases: (1) distance (or angle) estimation and (2) distance (or angle) combining. The most popular methods for estimating the distance between two nodes are:

- **Received Signal Strength Indicator (RSSI)** techniques measure the power of the signal at the receiver. Based on the known transmit power, the effective propagation loss can be calculated. Theoretical and empirical models are used to translate this loss into a distance estimate. This method has been used mainly for RF signals.

- **Time based methods (ToA,TDoA)** record the time-of-arrival (ToA) or time-difference-of-arrival (TDoA). The propagation time can be directly translated into distance, based on the known signal propagation speed. These methods can be applied to many different signals, such as RF, acoustic, infrared and ultrasound.
- **Angle -of -Arrival (AoA)** systems estimate the angle at which signals are received and use simple geometric relationships to calculate node positions.

A detailed discussion of these methods can be found in [20]. For the combining phase, the most popular alternatives are:

- The most basic and intuitive method is called hyperbolic tri-lateration. It locates a node by calculating the intersection of 3 circles (figure 3a).
- Triangulation is used when the direction of the node instead of the distance is estimated, as in AoA systems. The node positions are calculated in this case by using the trigonometry laws of sines and cosines (figure 3b).
- The third method is Maximum Likelihood (ML) estimation (figure 3c). It estimates the position of a node by minimizing the differences between the measured distances and estimated distances. We have chosen this technique as the basis of AHLoS for obtaining the Minimum Mean Square Estimate(MMSE) from a set of noisy distance measurements.

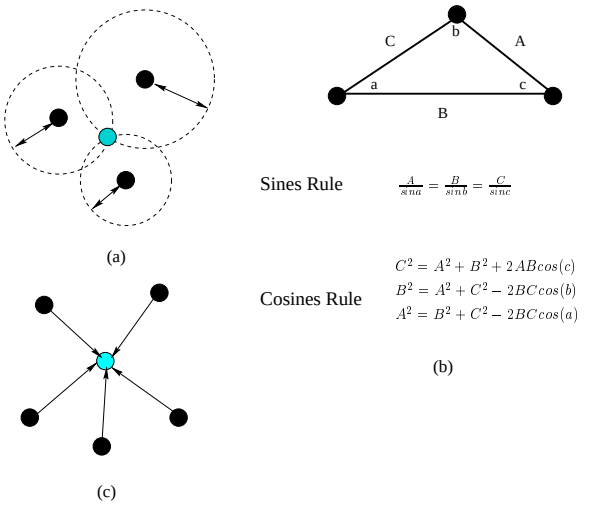


Figure 3: Localization Basics a) Hyperbolic trilateration, b) Triangulation, c) ML Multilateration

2.2 Related Work

In the past few decades, numerous localization systems have been developed and deployed. In the 1970s, the automatic vehicle location (AVL) systems were deployed to determine the position of police cars and military ground transportation vehicles. A set of stationary base stations acting as observation points use ToA and TDoA techniques to generate distance estimates. The vehicle position is then derived through multilaterations, using Taylor Series Expansion to transform a non-linear least squares problem to a linear [7][8]. Similar approaches can also be found in military applications for determining the position of airplanes.

In 1993, the well-known Global Positioning System (GPS) [34] system was deployed, which is based on the NAVSTAR satellite constellation (24 satellites). LORAN [28] operates in a similar way to GPS but uses ground based beacons instead of sattelites. In 1996, the U.S Federal Communications Commission (FCC) required that all wireless service providers give location information to the Emergency 911 services. Cellular base stations are used to locate mobile telephone users within a cell [9][10]. Distance estimates are generated with TDoA. The base station transmits a beacon and the handset reflects the signal back to the base station. Location information is again calculated by multilateration using least squares methods. By October 2001, FCC requires a 125-meter root mean square(RMS) accuracy in 67% of the time and by October 2006 a 300-meter RMS accuracy for 95% of the times is required.

Recently, there has been an increasing interest for indoor localization systems. The RADAR system [1] can track the location of users within a building. To calculate user locations the RADAR system uses RF signal strength measurements from three fixed base stations in two phases. First, a comprehensive set of received signal strength measurements is obtained in an offline phase to build a set of signal strength maps. The second phase is an online phase during which the location of users can be obtained by observing the received signal strength from the user stations and matching that with the readings from the offline phase. This process, eliminates multipath and shadowing effects at the cost of considerable preplanning effort.

The Cricket location support system [4] is also designed for indoor localization. It provides support for context aware applications and is low cost. Unlike the systems discussed so far, it uses ultrasound instead of RF signals. Fixed beacons inside the building distribute geographic information to the listener nodes. Cricket can achieve a granularity of 4 by 4 feet. Room level granularity can be obtained by the active badge [22] system, which uses infrared signals. The next development in this area on indoor localization is BAT [29] [30]. A BAT node carries an ultrasound transmitter whose signals are picked up by an array of receivers mounted on the ceiling. The location of a BAT can be calculated via multilateration with a few centimeters of accuracy. An RF base station coordinates the ultrasound transmissions such that interference from nearby transmitters is avoided. This system relies heavily on a centralized infrastructure.

In the ad-hoc domain, fewer localization systems exist. An RF based proximity method is presented in [21], in which the location of a node is given as a centroid. This centroid is generated by counting the beacon signals transmitted by a set of beacons pre-positioned in a mesh pattern. A different approach is taken in the Picoradio project at UC Berkeley. It provides a geolocation scheme for an indoor environment [11], based on RF received signal strength measurements and pre-calculated signal strength maps.

Our system, AHLoS, also belongs to the ad-hoc class. Although uses RF and ultrasound transmissions similar to the

Cricket and BAT Systems, it also has some key differences. AHLoS does not rely on a preinstalled infrastructure. Instead, it is a fully ad-hoc system with distributed localization algorithms running at every node. This results in a flexible system that only requires a small initial fraction of the nodes to be aware of their locations. Furthermore, it enables nodes to estimate their locations even if they are not within range with the beacon nodes. From a power awareness perspective, it also ensures that all nodes play an equal role in the location discovery process resulting in an even distribution of power consumption. The resulting localization system provides fine-grained localization with an accuracy of a few centimeters, similar to the BAT system without requiring infrastructure support. Finally, unlike all the systems discussed so far, AHLoS provisions for dynamic on-line estimation of the ultrasound propagation characteristics. This renders our approach extremely robust even in the presence of changing environments.

3. RESEARCH METHODOLOGY

As a first step in our study, we characterize the *ranging* capabilities of our two target technologies: Received RF signal strength using the WINS nodes and RF and ultrasound ToA using the *Medusa* nodes.

3.1 Ranging Characterization

3.1.1 Received Signal Strength

The signal strength method uses the relationship of RF signal attenuation as a function of distance. From this relationship a mathematical propagation model can be derived. From detailed studies of the RF signal propagation characteristics[18], it is well known that the propagation characteristics of radio signals can vary with changes in the surrounding environment (weather changes, urban / rural and indoor / outdoor settings). To evaluate signal strength measurements we conducted some experiments with the target system of interest, the WINS sensor nodes [12]. The WINS nodes have a 200MHz StrongARM 1100 processor, 1MB Flash, 128KB RAM and the Hummingbird digital cordless telephony (DECT) radio chipset that can transmit at 15 distinct power levels ranging from -9.3 to 15.6 dBm (0.12 to 36.31 mW). The WINS nodes carry an omni-directional antenna hence the radio signal is uniformly transmitted with the same power in all directions around the node. As part of the radio architecture, the WINS nodes provide a pair of RSSI (Received Signal Strength Indicators) resisters. RSSI registers are a standard feature in many wireless network cards [23]. Using these registers we conducted a set of measurements in order to derive an appropriate model for ranging. We performed measurements in several different settings (inside our lab, in the parking lot and between buildings). Unfortunately, a consistent model of the signal attenuation as a function of distance could not be obtained. This is mainly attributed to multipath, fading and shadowing effects. Another source of inconsistency is the great variation of RSSI associated with the altitude of the radio antenna. For instance, at ground level, the radio range at the maximum transmit power level the usable radio transmission range is around 30m whereas when the node is placed at a height of 1.5m the usable transmission range increases to around 100m. Because of these inconsistencies, we were only able to derive a model for an idealized setting; in a football field with all the nodes positioned at ground level. For this setup we developed a model based on the RSSI register readings at different transmission power levels and different node separations.

A model (equation 1) is derived by obtaining a least square fit for each power level. P_{RSSI} is the RSSI register reading and r is the distance between two nodes. Parameters X and n are constants that can be derived as functions of distance r for each power level. Averaged measurements and the corresponding derived models are shown in figure 4. Table 1 gives the X and n parameters for each case.

$$P_{RSSI} = \frac{X}{r^n} \quad (1)$$

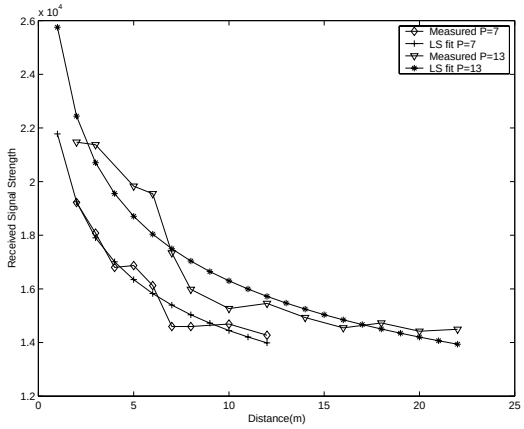


Figure 4: Radio Signal Strength Radio Characterization using WINS nodes(power levels P=7,13)

Table 1: RSSI Ranging Model Parameters for WINS nodes

Power Level	dBm	mW	X	n
7	2.5	1.78	21778.338	0.178186
13	14.4	27.54	25753.63	0.198641

With all the nodes placed on a flat plane, signal strength ranging can provide a distance estimate with an accuracy of a few meters. In all other cases, this experiment has shown that the use of radio signal strength can be very unpredictable. Another problem with the received signal strength approach is that radios in sensor nodes are low cost ones without precise well-calibrated components, such as the DECT radios in Rockwell's nodes or the emerging Bluetooth radios. As a result, it is not unusual for different nodes to exhibit significant variation in actual transmit power for the same transmit power level, or in the RSSI measured for the same actual received signal strength. Differences of several dBs are often seen. While these variations are acceptable for using transmit power adaptation and RSSI measurements for link layer protocols, they do not provide the accuracy required for fine-grained localization. A potential solution

would be to calibrate each node against a reference node prior to deployment, and store gain factors in non-volatile storage so that the run-time RSSI measurements may be normalized to a common scale.

3.1.2 ToA using RF and Ultrasound

To characterize ToA ranging on the *Medusa* nodes we measure the time difference between two simultaneously transmitted radio and ultrasound signals at the receiver (figure 5).

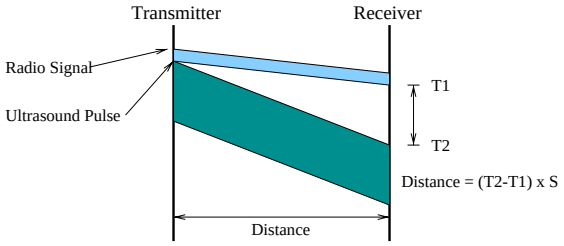


Figure 5: Distance measurement using ultrasound and radio signals

The ultrasound range on the *Medusa* nodes is about 3 meters (approximately 11-12 feet). We found this to be a convenient range for performing multihop experiments in our lab but we note that longer ranges are also possible at higher cost and power premiums. The Polaroid 6500 ultrasonic ranging module [17] for example has a range of more than 10 meters (the second generation of *Medusa* nodes will have a 10-15 meter range). We characterize ToA ranging by using two *Medusa* nodes placed on the floor of our lab. We recorded the time difference of arrival at 25-centimeter intervals. The results of our measurements are shown in figure 6. The x axis represents distance in centimeters and the y axis represent the microcontroller timer counter value.

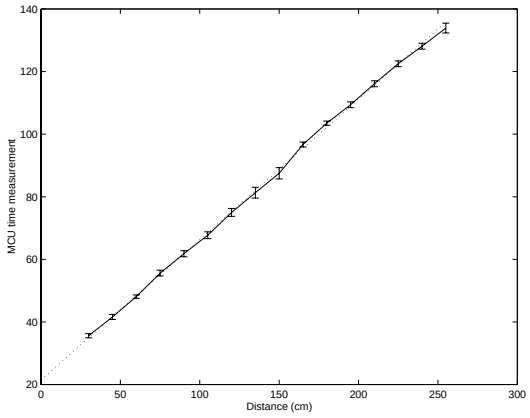


Figure 6: Ulrasound Ranging Characterization

The speed of sound is characterized in terms of the microcontroller timer ticks. To estimate the speed to sound as a function of microcontroller time, we perform a best line fit using linear regression (equation 2). s is the speed of sound in timer ticks, d is the estimated distance between 2 nodes and k is a constant. For this model $s = 0.4485$ and $k = 21.485831$.

$$t = sd + k \tag{2}$$

This *ranging* system can provide an accuracy of 2 centimeters for node separations under 3 meters. Like the RF signals, ultrasound also suffers from multipath effects. Fortunately, they are easier to detect. ToA measurement use the first pulse received ensuring that the shortest path(straight line) reading is observed. Reflected pulses from nodes that do not have direct line of sight are filtered out using statistical techniques similar to the ones used in [30].

3.2 Signal Strength vs. ToA ranging

On comparing the two ranging alternatives, we found that ToA using RF and ultrasound is more reliable than received signal strength. While received signal strength is greatly affected by amplitude variations of the received signal, ToA ranging only depends on the time difference, a much more robust metric. Based on our characterization results we chose ToA as the primary ranging method for AHLoS. Similar to RF signals, the ultrasound signal propagation characteristics may change with variations in the surrounding environment. To minimize these effects, AHLoS dynamically estimates the signal propagation characteristics every time sufficient information is available. This ensures that AHLoS will operate in many diverse environments without prior calibration. If the sensor network is deployed over a large field, the signal propagation characteristics may vary from region to region across the field. The calculation of the ultrasound propagation characteristics in the locality of each node ensures better location estimates accuracy. Table 2 summarizes the comparison between signal strength and ultrasound ranging. One possible solution we are considering for our future work is to combine received signal strength and ToA methods. Since the received signal strength method has the same effective range as the radio communication range, it can be used to provide a proximity indication in places where the network connectivity is very sparse for ToA localization to take place. The ultrasound approach will provide fine grained localization in denser parts of the networks. For this configuration, we plan to have the *Medusa* boards act as *location coprocessors* for the WINS nodes.

4. LOCALIZATION ALGORITHMS

Given a *ranging* technology that estimates node separation we now describe our localization algorithms. These algorithms operate on an ad-hoc network of sensor nodes where a small percentage of the nodes are aware of their positions either through manual configuration or using GPS. We refer to the nodes with known positions as *beacon* nodes and those with unknown positions as *unknown* nodes. Our goal is to estimate the positions of as many *unknown* nodes as possible in a fully distributed fashion. The proposed location discovery algorithms follow an iterative process. After the sensor network is deployed, the *beacon* nodes broadcast their locations to their neighbors. Neighboring *unknown* nodes measure their separation from their neighbors and use the broadcasted *beacon* positions to estimate their own positions. Once an *unknown* node estimates its position, it becomes a *beacon* and broadcasts its estimated position to other nearby *unknown* nodes, enabling them to estimate their positions. This process repeats until all the *unknown*

Table 2: A comparison of RSSI and ultrasound ranging

Property	RSSI	Ultrasound
Range	same as radio communication range	3 meters (up to a few 10s of meters)
Accuracy	O(m), 2-4m for WINS	O(cm), 2cm for Medusa
Measurement Reliability	hard to predict, multipath and shadowing	multipath mostly predictable,time is a more robust metric
Hardware Requirements	RF signal strength must be available to CPU	ultrasound transducers and amplifier circuitry
Additional Power Requirements	none	tx and rx signal amplification
Challenges	large variances in RSSI readings, multipath, shadowing, fading effects	interference, obstacles, multipath

nodes that satisfy the requirements for multilateration obtain an estimate of their position. This process is defined as *iterative multilateration* which uses *atomic multilateration* as its main primitive. In the following subsections we provide the details of atomic and iterative multilateration. Furthermore, we describe *collaborative multilateration* as an additional enhanced primitive for iterative multilateration and we provide some suggestions for further optimizations.

4.1 Atomic Multilateration

Atomic multilateration makes up the basic case where an *unknown* node can estimate its location if it is within range of at least three beacons. If three or more beacons are available, the node also estimates the ultrasound speed of propagation for its locality. Figure 7a illustrates a topology for which atomic multilateration can be applied.

The error of the measured distance between an unknown node and its ith beacon can be expressed as the difference between the measured distance and the estimated Euclidean distance (equation 3). x_0 and y_0 are the estimated coordinates for the unknown node 0 for $i = 1, 2, 3...N$, where N is the total number of beacons, and t_{i0} is the time it takes for an ultrasound signal to propagate from beacon i to node 0, and s is the estimated ultrasound propagation speed.

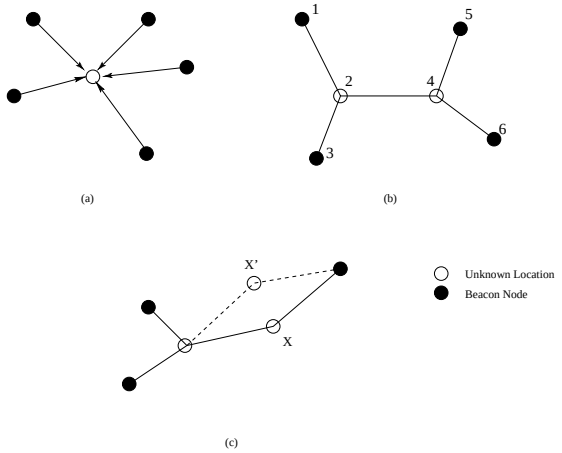


Figure 7: Multilateration examples

$$f_i(x_0, y_0, s) = st_{i0} - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \qquad (3)$$

Given that an adequate number of beacon nodes are available, a Maximum Likelihood estimate of the node's position can be obtained by taking the minimum mean square estimate (MMSE) of a system of $f_i(x_0, y_0, s)$ equations (equation 4). Term α represents the weight applied to each equation. For simplicity we assume that $\alpha = 1$.

$$F(x_0, y_0, s) = \sum_{i=1}^{N} \alpha^2 f(i)^2 \qquad (4)$$

If a node has three or more beacons a set of three equations of the form of (3) can be constructed yield an overdetermined system with a unique solution for the position of the unknown node 0. If four or more beacons are available, the ultrasound propagation speed s can also be estimated. The resulting system of equations can be linearized by setting $f_i(x_0, y_0, s)$ = equation 3, squaring and rearranging terms to obtain equation 5.

$$\begin{aligned} -x_i^2 - y_i^2 = \\ (x_0^2 + y_0^2) + x_0(-2x_i) + y_0(-2y_i) - s^2 t_{i0}^2 \end{aligned} \qquad (5)$$

for k such equations we can eliminate the $(x_0^2 + y_0^2)$ terms by subtracting the kth equation from the rest.

$$\begin{aligned} -x_i^2 - y_i^2 + x_k^2 + y_k^2 = 2x_o(x_k - x_i) \\ +2y_0(y_k - y_i) + s^2({t_{ik}}^2 - {t_{i0}}^2) \end{aligned} \qquad (6)$$

this system of equations has the form $y = bX$ and can be solved using the matrix solution for MMSE [25] given by $b = (X^T X)^{-1} X^T y$ where

$$X = \begin{bmatrix} 2(x_k - x_1) & 2(y_k - y_1) & {t_{k0}}^2 - {t_{k1}}^2 \\ 2(x_k - x_2) & 2(y_k - y_2) & {t_{k0}}^2 - {t_{k2}}^2 \\ \vdots & \vdots & \vdots \\ 2(x_k - x_{k-1}) & 2(y_k - y_{k-1}) & {t_{k0}}^2 - {t_{k(k-1)}}^2 \end{bmatrix}$$

$$y = \begin{bmatrix} -x_1^2 - y_1^2 + x_k^2 + y_k^2 \\ -x_2^2 - y_2^2 + x_k^2 + y_k^2 \\ \vdots \\ x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 \end{bmatrix}$$

and

$$b = \begin{bmatrix} x_0 \\ y_0 \\ s^2 \end{bmatrix}$$

Based on this solution we define the following requirement for atomic multilateration.

REQUIREMENT 1. *Atomic multilateration can take place if the unknown node is within one hop distance from at least three beacon nodes. The node may also estimate the ultrasound propagation speed if four or more beacons are available.*

Although requirement 1 is necessary for atomic multilateration, it is not always sufficient. In the special case when beacons are in a straight line, a position estimate cannot be obtained by atomic multilateration. If this occurs, the node will attempt to estimate its position using collaborative multilateration. We also note that atomic multilateration can be performed in 3-D without requiring an additional beacon [33].

4.2 Iterative Multilateration

The *iterative multilateration* algorithm uses atomic multilateration as its main primitive to estimate node locations in an ad-hoc network. This algorithm is fully distributed and can run on each individual node in the network. Alternatively, the algorithm can also run at a single central node or a set of cluster-heads, if the network is cluster based. Figure 8 illustrates how iterative multilateration would execute from a central node that has global knowledge of the network. The algorithm operates on a graph G which represents the network connectivity. The weights of the graph edges denote the separation between two adjacent nodes. The algorithm starts by estimating the position of the unknown node with the maximum number of beacons using atomic multilateration. Since at a central location all the the entire network topology is known so we start from the *unknown* node with the maximum number of beacons to obtain better accuracy and faster convergence (in the distributed version an *unknown* will perform a multilateration as soon as information from three beacons). When an *unknown* node estimates its location, it becomes a *beacon*. This process repeats until the positions of all the nodes that eventually can have three or more beacons are estimated.

```
boolean iterativeMultilateration (G)
  (MaxBeaconNode, BeaconCount) ← unknown
    node with most beacons
  while BeaconCount ≥ 3
    setBeacon (MaxBeaconNode)
    (MaxBeaconNode, BeaconCount) ← unknown
    node with most beacons
```

Figure 8: Iterative Multilateration Algorithm as it executes on a centralized node

A drawback of iterative multilateration is the error accumulation that results from the use of *unknown* nodes that estimate their positions as *beacons*. Fortunately, this error accumulation is not very high because of the high precision of our ranging method. Figure 9 shows the position errors in a simulated network of 50 *Medusa* nodes when 10% of the nodes are initially configured as beacons. The nodes are deployed on a square grid of side 15 meters. The simulation considers two types of errors: 1) ranging errors and 2) beacon placement errors. In both cases a 20mm white Gaussian error is used. In both cases the estimated node positions are within 20 cm from the actual positions.

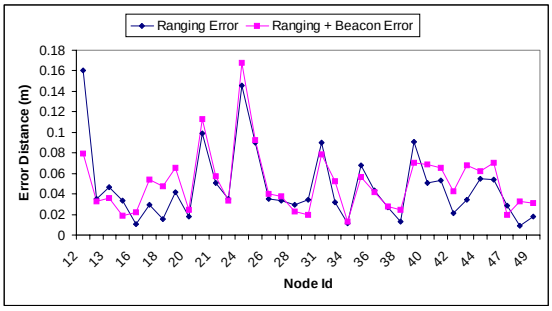


Figure 9: Iterative Multilateration Accuracy in a network of 50 nodes and 10% beacons

4.3 Collaborative Multilateration

In an ad-hoc deployment with random distribution of *beacons*, it is highly possible that at some nodes, the conditions for atomic multilateration will not be met; i.e. an *unknown* node may never have three neighboring beacon nodes therefore it will not be able to estimate its position using *atomic multilateration*. When this occurs, a node may attempt to estimate its position by considering use of location information over multiple hops in a process we refer to as *collaborative multilateration*. If sufficient information is available to form an over-determined system of equations with a unique solution set, a node can estimate its position and the position of one or more additional unknown nodes by solving a set of simultaneous quadratic equations. Figure 7b illustrates one of the most basic topologies for which *collaborative multilateration* can be applied. Nodes 2 and 4 are *unknown* nodes, while nodes 1,3,5,6 are *beacon* nodes. Since both nodes 2 and 4 have three neighbors (degree $d = 3$) and all the other nodes are *beacons*, a unique position estimate for nodes 2 and 4 can be computed. More formally, collaborative multilateration can be stated as follows: Con-

sider the ad-hoc network to be a connected undirected graph $G = (N, E)$ consisting of $|N| = n$ nodes and a set E of $n-1$ or more edges. The *beacon* nodes are denoted by a set B where $B \subseteq N$ and the set of *unknown* nodes is denoted by U where $U \subseteq G$. Our goal is to solve for

$x_u, y_u \ \forall u \subseteq U$ by minimizing

$$f(x_u, y_u) = D_{iu} - \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2} \qquad (7)$$

for all participating node pairs i, u where $i \subseteq B$ or $i \subseteq U$ and $u \subseteq U$. Subject to:
x_i, y_i are known if $i \subseteq B$, and every node pair i, u is a participating pair. Participating nodes and participating pair are defined as follows:

DEFINITION 1. *A node is a participating node if it is either a beacon or if it is an unknown with at least three participating neighbors.*

In figure 7b if collaborative multilateration starts at node 2, node 2 must have at least three participating neighbors. Nodes 1 and 3 are beacons therefore they are participating. Node 4 is unknown but has two beacons: nodes 5 and 6. Node 4 is also connected to node 2 thus making both of them participating nodes.

DEFINITION 2. *A participating node pair is a beacon-unknown or unknown-unknown pair of connected nodes where all unknowns are participating.*

In this formulation, the nodes participating in collaborative multilateration make up a subgraph of G, for which an equation of the form of 7 can be written for each edge E that connects a pair of participating nodes. To ensure a unique solution, all nodes considered must be participating. In figure 7b for example, we have five edges thus a set of five equations can be obtained. In some cases other cases , we may have a well-determined system of n equations and n *unknowns* such as in the case shown figure 7c. We can easily observe however, that node X can have two possible positions that would satisfy this system therefore the solution is not unique and node X is not a participating node. If the above conditions are met, the resulting system of nonlinear equations can be solved with optimization methods such as gradient descend [26] and simulated annealing [27].

The algorithm in figure 10 provides a basic example of how a node determines whether it can initiate collaborative multilateration. The parameter *node* denotes the node id from where the search for a collaborative multilateration begins. The second parameter *callerId* holds the node id of the node that calls the particular instance of the function. *isInitiator* is a boolean variable that is set to *true* if the node was the initiator of the collaborative multilateration process and *false* otherwise. This is used to set the *limit* flag that drives the recursion.

```
boolean isCollaborative (node, callerId, isInitiator)
  if isInitiator==true limit ← 3
  else limit ← 2
  count ← beaconCount(node)
  if count ≥ limit return true
  for each unknown neighbor i not previously visited
    if isCollaborative (i, node, false) count++
    if count == limit return true
  return false
```

Figure 10: Algorithm for checking the feasibility for collaborative multilateration

Collaborative multilateration can be used to assist iterative multilateration in places of the network where the beacon density is low and the requirement for atomic multilateration is not satisfied. Figure 11 illustrates how iterative multilateration would call collaborative multilateration when the requirement for atomic multilateration is not met.

```
boolean iterativeMultilateration (G)
   (MaxBeaconNode, BeaconCount) ← unknown
     node with most beacons
  while BeaconCount ≥ 3
    setBeacon (MaxBeaconNode)
   (MaxBeaconNode, BeaconCount) ← unknown
     node with most beacons
    while isCollaborative (MaxBeaconNode, -1, true)
      set all nodes in collaborative set as beacons
     (MaxBeaconNode, BeaconCount) ← unknown
       node with most beacons
      while BeaconCount ≥ 3
        setBeacon (MaxBeaconNode)
       (MaxBeaconNode, BeaconCount) ← unknown
         node with most beacons
```

Figure 11: Enhanced Iterative Multilateration

Collaborative multilateration can help in situations where the percentage of beacons is low. This effect is shown in figure 12. This scenario considers a sensor field of 100 by 100 and a sensing range of 10 and two network sizes of 200 and 300 nodes. As shown in the figure, if the percentage of *beacons* is small, the number of node locations that can be resolved is substantially increased with collaborative multilateration. This result also shows how network density is related to the localization process. In the 300 node network, more node locations can be estimated than in the 200 node network with the same percentage of beacons. This is due to the higher degree of connectivity. The effects of node and beacon placement on the localization process is studied in more detail in section 5.

4.4 Further Optimizations

The accuracy of the estimated locations in the multilateration algorithms described in this section may be further improved with two additional optimizations. First, error

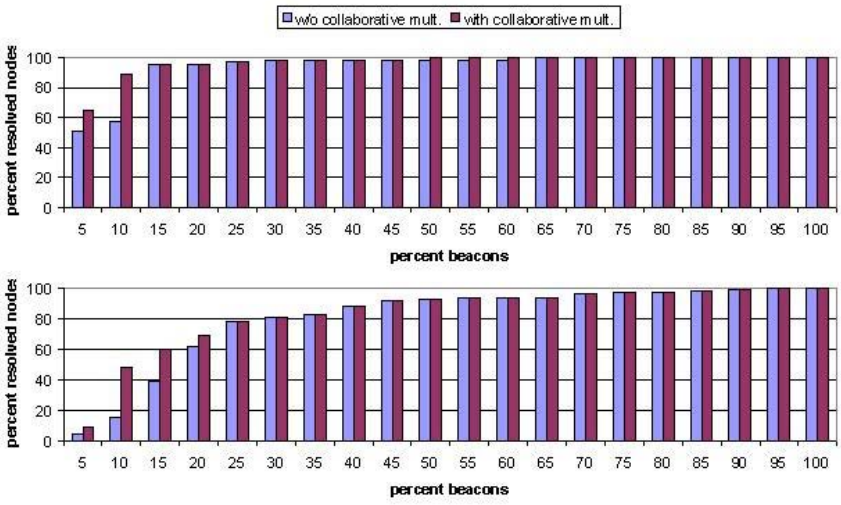


Figure 12: Effect of collaborative multilateration top, 300 nodes, bottom 200 nodes

propagation can be reduced by using weighted multilateration. In this scheme, beacons with higher certainty about their location are weighted more than beacons with lower certainty during a multilateration. As new nodes become beacons, the certainty of their estimated location can also be computed and used as a weight in future multilaterations. Additionally, by applying collaborative multilateration over a wider scope, the accumulated error can be reduced. The solution methodology and further evaluation of these optimizations are part of our future work and will be the subject of a future paper.

5. NODE AND BEACON PLACEMENT

The success of the location discovery algorithm depends on network connectivity and beacon placement. In this section, we conduct a brief probabilistic analysis to determine how the connectivity requirements can be met when nodes are uniformly deployed in a field. Based on these results, we later perform a statistical analysis to get an indication on the percentage of beacons required. When considering node deployment, the main metric of interest is the probability with which any node in the network has a degree of three or more, assuming that sensor nodes are uniformly distributed over the sensor field. In a network of N nodes deployed in a square field of side L, the probability $P(d)$ of a node having degree d is given by the binomial distribution in equation 8 and the probability P_R being in transmission range.

$$P(d) = P_R^d.(1-P_R)^{N-d-1}.\binom{N-1}{d} \tag{8}$$

$$P_R = \frac{\pi R^2}{L^2} \tag{9}$$

For large values of N tending to infinity, the above binomial distribution converges to a Poisson distribution. When taking into account that $\lambda = N.P_R$ we get equation 10, the probability of a node have degree of three or more can be calculated. Also, an indication of the number of nodes required per unit area can be calculated in terms of λ. Table 3 shows the number of nodes required to cover a square field of size $L = 100$ and range $R = 10$ as well as the probability for a node to have degree greater than three or four for different values of λ. These probabilities are obtained from equation 11.

$$P(d) = \frac{\lambda^d}{d!}.e^{-\lambda} \tag{10}$$

$$P(d \geq n) = 1 - \sum_{i=0}^{n-1} P(i) \tag{11}$$

Table 3: Probability of node degree for different λ values

λ	P($d \geq 3$)	P($d \geq 4$)	nodes/10,000m^2
2	0.323324	0.142877	39
4	0.761897	0.56653	78
6	0.938031	0.848796	117
8	0.986246	0.95762	157
10	0.997231	0.989664	196
12	0.999478	0.997708	235
14	0.999906	0.999526	274
16	0.999984	0.999907	314
18	0.999997	0.999982	353
20	1	0.999997	392

The connectivity results in figure 13 show the probabilities of a node having 0,1,2 or 3 and more neighbors. In addition to node connectivity, we determine percentage of initial beacon nodes required for the convergence of the localization

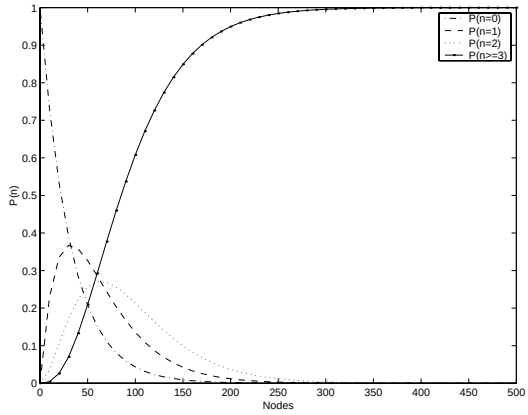


Figure 13: Connectivity result for a 100 x 100 field and sensor range 10

algorithm by statistical analysis. Using the same network setup as before, we report the percentage of nodes that estimate their locations while varying the percentage of nodes and beacons. The results in figure 14 are the averages over 100 simulations. The figure shows the percentage of beacons required to complete the iterative multilateration process using only atomic multilaterations. We note that the percentage of required beacons decreases as network density increases. Also as the network density increases, the transitions in the required number of beacons become much sharper since the addition of a few more beacon nodes reinforces the progress of the iterative multilateration algorithm.

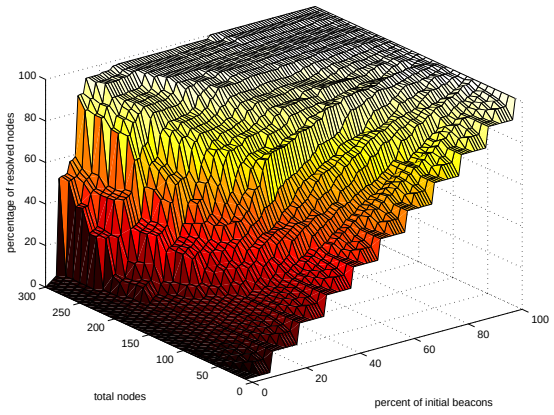


Figure 14: Beacon Requirements for different node densities

6. IMPLEMENTATION AND EXPERIMENTATION

6.1 Medusa Node Architecture

The *Medusa* node design (figure 2) is based on the AVR 8535 processor [13] which carries 8KB of flash memory, 512 bytes SRAM and 512 bytes of EEPROM memory. The radio we use is the DR3000 radio module from RF Monolithics[14]. This radio supports two data rates (2.4 and 19.2 kbps) and two modulation schemes (ASK and OOK). The ultrasound circuitry consists of six (60 degree detect angle) pairs of 40KHz ultrasonic transducers arranged in a hexagonal pattern at the center of the board (note that for experimental purposes the *Medusa* node in figure 2 has 8 transducers). Each ultrasound transceiver is supported by a pair of solid core wires at an approximate height of 15 cm above the printed circuit board. We found this very convenient setup for experimentation since it allows the transceivers to be rotated in different directions. The first generation board is 3" x 4" and it is powered by a 9V battery. The *Medusa* firmware is based on an event driven firmware implementation suggested in [15]. The radio communication protocols use a variable size framing scheme, 4-6 bit encoding [16] and 16 bit CRC. The code for *ranging* is integrated in the ad-hoc routing protocol described in the next subsection.

6.2 Location Information Dissemination and Routing

In our experimental setup all measurements from the nodes are forwarded to a PC basestation. To route messages to the base station, we implemented a lightweight version of the DSDV [19] routing algorithm, which we refer to as DSDVlite. Instead of maintaining a routing table with the next hop information to every node, DSDVlite only maintains a short routing table that holds next hop information for the shortest route to gateway. Furthermore, this algorithm drives the localization process by carrying the location information of beacons, and by ensuring that the received ultrasound beacon signals originate from the same source node as the radio signals. The ultrasound beacon signal transmission begins right after the transmission of the start symbol for each routing packet. After this, the transmission of data and ultrasound signals proceed simultaneously. By ensuring that the duration of the data transmission is longer than the ultrasound transmission, the receiver can differentiate between erroneous ultrasound transmissions from other nodes. If the data packet is not correctly received because of a collision with another transmission, it also implies a collision of ultrasound signals hence the ultrasound time measurement is discarded.

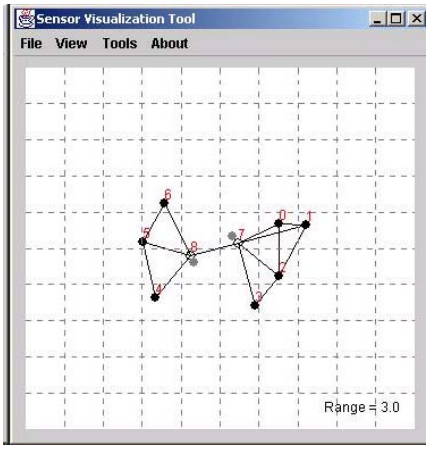


Figure 15: 9 node scenario

6.3 Experimental Setup

Our experimental testbed consists of 9 *Medusa* nodes and a Pentium II 300MHz PC. One node is configured as a gateway and it is attached to the PC through the serial port. Some of the nodes are pre-programmed with their locations and they act as beacons. All the nodes perform ranging and they transmit all the ranging information to the PC that runs the localization algorithms and displays the node positions on a sensor visualization tool. The node positions on the sensor visualization tool are updated at 5-second intervals. Figures 16 and 15 show some snapshots of node locations. The beacons are shown as black dots, the unknown nodes are white circles and the node position estimates are shown as gray dots. In all of our experiments all the node position estimates for each unknown node always fall within the 3" x 4" surface area of the *Medusa* boards.

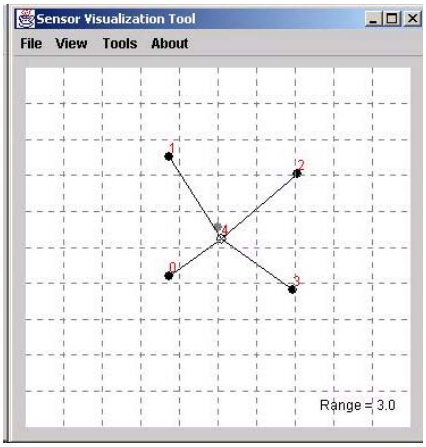


Figure 16: 5 node scenario

6.4 Power Characterization

In the previous subsection we verified the correct operation of our localization system. Our experimental setup will provide a reasonable solution for a small network but as the network scales, the traffic to the central gateway node will increase substantially. Before we can evaluate the tradeoffs between estimating locations at the nodes and estimating locations at a central node we first characterize power consumption of the *Medusa* nodes at different operational modes. Using an HP 1660 Logic Analyzer, a bench power supply and a high precision resistor we characterized the RFM radio and the AVR microcontroller on the *Medusa* nodes.

$$I_{sensor} = \frac{V_{test}}{R_{test}}$$

$$V_{Sensor} = V_{supply} - V_{test}$$

$$Power = I_{sensor} \times V_{sensor}$$

$$Energy = Power \times time$$

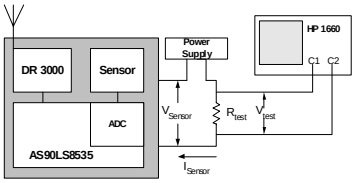


Figure 17: Power and Energy Relationships and Measurement Setup

The measurement setup and power/energy relationships are shown in figure 17. The power consumption for different modes of the AVR microcontroller are shown in table 4. The power consumption for the different modes of the RFM radio are shown in figure 18 and table 5.

Table 4: AVR 8535 Power Characterization

AVR Mode	Current	Power
Active	2.9mA	8.7mW
Sleep	1.9mA	5.9mW
Power Down	1μA	3μW

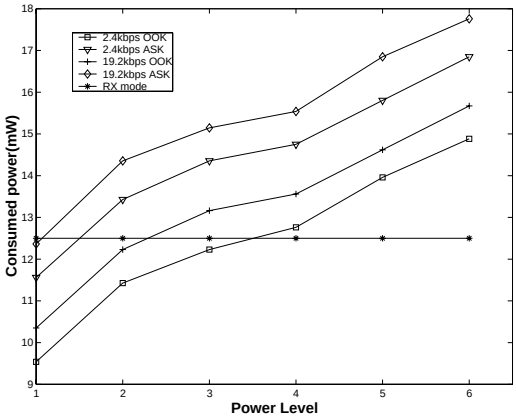


Figure 18: RFM Radio power consumption at different operational modes

7. TRADEOFFS BETWEEN CENTRALIZED AND DISTRIBUTED SCHEMES

One important aspect that needs to be determined is whether the location estimation should be done in a centralized or distributed fashion. In the former case, all the ranging measurements and beacon locations are collected to a central base station where the computation takes place and the results are forwarded back to the nodes. In the latter, each node estimates its own location when the requirements for atomic multilateration are met. For the AHLoS system, we advocate that distributed computation would be a better choice since a centralized approach has several drawbacks. First, to forward the location information to a central node, a route to the central node must be known. This implies the use of a routing protocol other than location based routing and also incurs some additional communication cost which is also affected by the efficiency of the existing routing and media access control protocols. Second, a centralized approach, creates a time synchronization problem. Whenever there is a change in the network topology the node's knowledge of location will not instantaneously updated. To correctly keep track of events, the central node will need to cache node locations to ensure consistency of event reports in space and time. Third, the placement of the central node implies some preplanning to ensure that the node is easily accessible by other nodes. Also, because of the large volume of traffic to and from the central node, the battery lifetime of the nodes around the central node will be seriously impacted. Fourth, the robustness of the system suffers. If the routes to the central node are broken, the nodes will

Table 5: RFM Power Characterization

Mode	Power Level	OOK Modulation				ASK Modulation			
		2.4Kbps		19.2Kbps		2.4Kbps		19.2Kbps	
	mW	mA	mW	mA	mW	mA	mW	mA	mW
Tx	0.7368	4.95	14.88	5.22	15.67	5.63	16.85	5.95	17.76
Tx	0.5506	4.63	13.96	4.86	14.62	5.27	15.80	5.63	16.85
Tx	0.3972	4.22	12.76	4.49	13.56	4.90	14.75	5.18	15.54
Tx	0.3307	4.04	12.23	4.36	13.16	4.77	14.35	5.04	15.15
Tx	0.2396	3.77	11.43	4.04	12.23	4.45	13.43	4.77	14.35
Tx	0.0979	3.13	9.54	3.40	10.35	3.81	11.56	4.08	12.36
Rx	-	4.13	12.50	4.13	12.50	4.13	12.50	4.13	12.50
Idle	-	4.08	12.36	4.08	12.36	4.08	12.36	4.08	12.36
Sleep	-	0.005	0.016	0.005	0.016	0.005	0.016	0.005	0.016

not be able to communicate their location information to the central node and vice versa. Finally, since all the raw data is required, the data aggregation that can be performed within the network to conserve communication bandwidth is minimal. One advantage of performing the computation at a centralized location is that more rigorous localization algorithms can be applied such as the one presented in [35]. Such algorithms however require much more powerful computational capabilities than the ones available at low cost sensor nodes. Overall, a centralized implementation will not only reduce the network lifetime but it will also increase its complexity and compromise its robustness. On the other hand, if location estimation takes place at each node in a distributed manner the above problems can be alleviated. Topology changes will be handled locally and the location estimate at each node can be updated at minimal cost. In addition, the network can operate totally on location based routing so the implementation complexity will be reduced. Also since each node is responsible for determining its location, the localization is more tolerant to node failures.

To evaluate energy consumption tradeoffs between the centralized and distributed approaches we run some simulations on a typical sensor network setup. In our scenario the central node is placed at the center of a square sensor field. Furthermore, we assume the use of an ideal, medium access control(MAC) and routing protocols. The MAC protocol is collision free and the routing protocol always uses the shortest route to the central node. The total number of bytes transmitted by all the nodes during both distributed and centralized localization is recorded. The network size varied with the network density kept constant by using a value of $\lambda = 6$ or 117 nodes for every $10{,}000m^2$ (from table 3). The simulation setup considers the same packet sizes as the implementation on the medusa nodes. For the centralized system each node forwards the range measurements between all its neighbors. If the node is *beacon* it also forwards its location information (this is 96 bits long which is equivalent to a GPS reading). Once the location is computed, the central node will forward the results back to node the corresponding unknown nodes. In the distributed setup, each node transmits a short beacon signal (radio and ultrasound pulse) followed by the senders location if the sender is a beacon. In both cases, the simulation runs for one full cycle of the localization process(until all feasible unknown node positions are resolved). The average number of transmitted bytes for each case are shown in figures 19 and 20 for 10% and 20% beacon density respectively. The results shown in the figure are averages of over 100 simulations with random node placement following a uniform distribution.

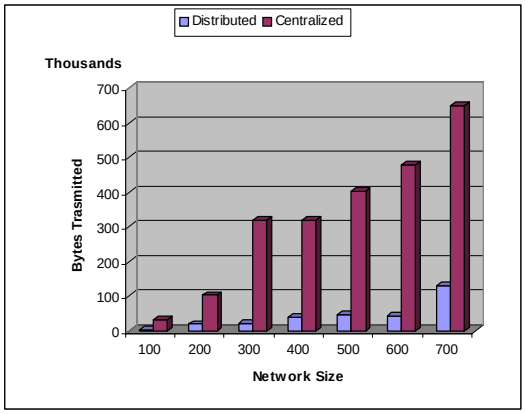


Figure 19: Traffic in distributed and centralized implementations with 10% beacons

Figure 21 shows the average energy consumption per node for the *Medusa* nodes when the radio transmission power is set to 0.24mW. This result is based on the power characterization of the *Medusa* nodes from the previous section. We also node that the energy overhead for the ultrasound based ranging is the same for both centralized and distributed schemes therefore it is not included in the energy results presented here. These results show that in the distributed setup has six to ten times less communiocation overhead than the centralized setup. Another interesting trend to note is that in the centralized setup, network traffic increases as the percentage of *beacon* nodes increases. In the distributed setup however, the traffic decreases as the percentage of *beacon* nodes increases. This decrease in traffic is mainly attributed to the fact that most of the times the localization process can converge faster if more beacon nodes are available; hence less information exchange has to take place between the nodes.

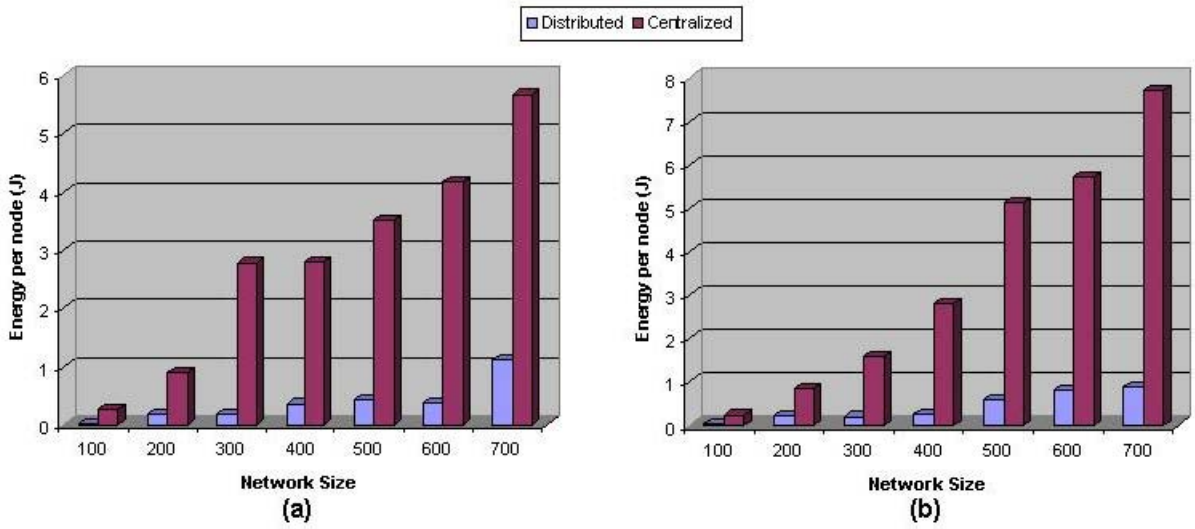


Figure 21: Average energy spent at a node during localization with a) 10% beacons, b) 20% beacons

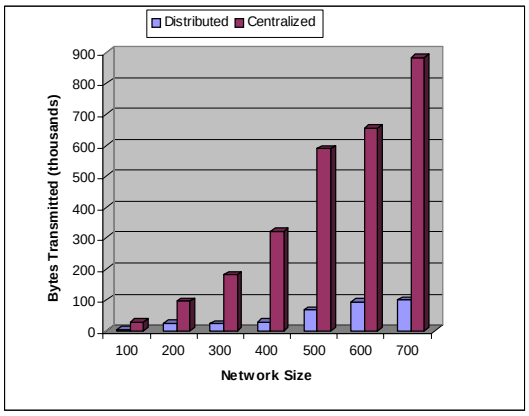


Figure 20: Traffic in distributed and centralized implementations with 20% beacons

8. CONCLUSIONS

We have presented a new localization scheme for wireless ad-hoc sensor networks. From our study we found that the use of ToA ranging is a good candidate for fine-grained localization as it is less sensitive to physical effects. Received RF signal strength ranging on the other hand is not suitable for fine-grained localization. Furthermore, we conclude that our fine-grained localization scheme should operate in a distributed fashion. Although more accurate location estimations can be obtained with centralized implementation, a distributed implementation will increase the system robustness and will result in a more even distribution of power consumption across the network during localization. Furthermore, the implementation of our testbed proved to be an indispensable tool for understanding and analyzing the strengths and limitations of our approach. Although our system performed very well for our experiments, we recommend the use of a more powerful CPU on the on the sensor nodes for the following reasons. First, RF and ultrasound ToA ranging requires the use of a dedicated high speed timer. In our implementation the 4MHz AVR microcontroller is dedicated to localization and this is sufficient. If however, the microcontroller is expected to perform additional tasks at the same time a higher performance processor is highly recommended. Based on our experience, we are currently developing a second generation of the *Medusa* nodes. These nodes will be capable of performing hybrid ranging by introducing the fusion of both ultrasonic ToA ranging and received signal strength RF ranging. Finally, in this initial study we found that the accuracy of iterative multilateration is satisfactory for small networks but needs to be improved for larger scale networks. To this end, as part of our future work we plan to extend our algorithms to achieve better accuracy by limiting the error propagation across the network.

Acknowledgments

This paper is based in part on research funded through NSF under grant number ANI-008577, and through DARPA SensIT and Rome Laboratory, Air Force Material Command, USAF, under agreement number F30602-99-1-0529. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF, DARPA, or Rome Laboratory, USAF. We also thank the anonymous reviews for their detailed feedback and suggestions.

9. REFERENCES

[1] P. Bahl, V. Padmanabhan, *RADAR: An In-Building RF-based User Location and Tracking System* Proceedings of INFOCOM 2000 Tel Aviv, Israel, March 2000, p775-84, vol. 2

[2] AVL Information Systems, Inc ,

http://www.avlinfosys.com/

[3] Deborah Estrin, Ramesh Govindan and John Heidemann, *Scalable Coordination in Sensor Networks* Proceedings of Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, WA, Aug 1999, p263-70

[4] N. Priyantha, A. Chakraborthy and H. Balakrishnan, *The Cricket Location-Support System*, Proceedings of International Conference on Mobile Computing and Networking, pp. 32-43, August 6-11, 2000, Boston, MA

[5] J.Li, J. Jannotti, D. S. J. DeCouto, D. R. Karger and R. Morris *A Scalable Location Service for Geographic Ad-Hoc Routing* Proceedings of ACM Mobile Communications Conference, August 6-11 2000, Boston, Massachusetts

[6] K. Amouris, S. Papavassiliou, M. Li *A Position-Based Multi-Zone Routing Protocol for Wide Area Mobile Ad-Hoc Networks*, Proceedings of VTC 99

[7] G. Turing, W. Jewell and T. Johnston, *Simulation of Urban Vehicle-Monitoring Systems* IEEE Transactions on Vehicular Technology, Vol VT-21, No1. Page 9-16, February 1972

[8] W. Foy *Position-Location Solution by Taylor Series Estimation* IEEE Transactions of Aerospace and Electronic Systems Vol. AES-12, No. 2, pages 187-193, March 1976

[9] J. Caffery and G. Stuber, *Subscriber Location in CDMA Cellular Networks* IEEE Transactions on Vehicular Technology, Vol. 47 No.2, pages 406-416, May 1998

[10] J. Caffery and G. Stuber, *Overview of Radiolocation in CDMA Cellular Systems* IEEE Communications Magazine, April 1999

[11] J. Beutel, *Geolocation in a PicoRadio Environment* Masters Thesis, UC Berkeley. July 1999.

[12] Wireless Intergated Network Systems(WINS) http://wins.rsc.rockwell.com/

[13] Atmel AS90LS8535, http://www.atmel.com/atmel/products/prod200.htm

[14] DR3000 ASH Radio Module, http://www.rfm.com/products/data/dr3000.pdf

[15] M. Melkonian, *Getting by without an RTOS* Embedded Systems Programming, September 2000

[16] RFM Software Designer's Guide, http://www.rfm.com/corp/apnotes.htm

[17] Polaroid 6500 ultrasonic ranging kit, http://www.acroname.com/robotics/parts/R11-6500.html

[18] T. Rappaport *Wireless Communications Principle and Practice* Prentice Hall, 1996

[19] C. Perkins and P. Bhaghwat, *Highly Dynamic Destination Sequenced Distance-Vector Routing* In proceedings of the SIGCOMM 94 Conference on Communication Architectures, Protocols and Applications, pages 234-244, August 1994.

[20] J. Gibson, *The Mobile Communications Handbook* IEEE Press 1999

[21] N. Bulusu, J. Heidemann and D. Estrin, *GPS-less Low Cost Outdoor Localization For Very Small Devices*, IEEE Personal Communications Magazine, Special Issue on Networking the Physical World, August 2000.

[22] R. Want, A. Hopper, V. Falcao and J. Gibbons, The Active Badge Location System. ACM Transactions on Information Systems 10, January 1992, pages 91-102

[23] WaveLAN White specs, www.wavelan.com products

[24] UCB/LBNL/VINT Network Simulator - ns (version 2) *http://www.isi.edu/nsnam/ns/*

[25] W. Greene, *Econometric Analysis,* Third Edition, Prentice Hall 1997

[26] D. J. Dayley and B. M. Bell, *A Method for GPS Positioning* IEE Trans., Aerosp. Electron. Syst., 1996, 32,(3),pp. 1148-54

[27] W. H. Press, et al. *Numerical recipes in C: the art of scientific computing, 2nd ed..* Cambridge ; New York: Cambridge University Press, 1992.

[28] *LORAN* http://www.navcen.uscg.mil/loran/Default.htm#Link

[29] A. Harter, A. Hopper, P. Steggles, A. Ward and P.Webster, *The Anatomy of a Context Aware Application* In Proceedings ACM/IEEE MOBICOM (Seattle, WA, Aug 1999)

[30] A. Harter, A. Hooper *A New Location Technique for the Active Office* IEEE Personal Communications vol 4,(No. 5), October 1997, pp. 42-47

[31] S. Meguerdichian , F. Koushanfar, M. Potkonjak and M. B. Srivastava *Coverage Problems in Wireless Ad-hoc Sensor Networks*, In proceedings of Infocom 2001, Ankorange, Alaska

[32] M. B. Srivastava , R. Muntz and M. Potkonjak, *Smart Kindergarten: Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments* In Proceedings of ACM/IEEE MOBICOM 2001

[33] D. E. Manolakis, *Efficient Solution and Performance Analysis of 3-D Position Estimation by Trilateration* IEEE Transactions on Aerospace and Electronic Systems vol 32, p1239-48, October 1996

[34] E. Kaplan, *Understanding GPS Principles and Applications* Artech House, 1996

[35] L. Doherty, K. Pister and L. E. Ghaoui, *Convex Optimization Methods for Sensor Node Position Estimation* Proceedings of INFOCOM 2001, Anchorage, Alaska, April 2001

Appendix B:

On Modeling Networks of Wireless Microsensors

Andreas Savvides, Sung Park and Mani Srivastava
Electrical Engineering Department
University of California, Los Angeles
{asavvide, spark, mbs}@ee.ucla.edu

1. INTRODUCTION

Recent advances in low-power embedded processors, radios, and micro-mechanical systems (MEMs) have made possible the development of networks of wirelessly interconnected sensors. With their focus on applications requiring tight coupling with the physical world, as opposed to the personal communication focus of conventional wireless networks, these wireless sensor networks pose significantly different design, implementation, and deployment challenges. Their application-specific nature, severe resource limitations, long network life requirements, and the presence of sensors lead to interesting interplay between sensing, communications, power consumption, and topology that designers need to consider. Existing tools for modeling wireless networks focus only on the communications problem, and do no support modeling the power and sensing aspects that are essential to wireless sensor network design. In this paper, we present a set of models and techniques that are embodied in a simulation tool [1] for modeling wireless sensor networks. Our models are derived with detailed power measurements involving 2 different types of sensor nodes representing two extremes; high-end WINS nodes [2] by Rockwell and low-end experimental nodes that we have built. The WINS nodes have a StrongARM S1100 processor and a 100m-range radio and can carry a wide variety of sensors. The experimental nodes feature an AVR 90LS8535 microcontroller from Atmel and a low power radio 20m-range from RFM Monolithics and a similar to the COTS nodes from UC Berkeley [3].

To instrument sensor network scenarios in a simulation environment, more features need are also introduced. The notion of a sensing channel is used to propagate stimuli to the sensors. Target models are responsible for generating the stimuli that trigger the sensors, which in turn become communication traffic towards a central base station. All these actions affect power consumption, which directly affect the useful lifetime of the network. Since power consumption is a crucial factor we focus our study on empirical measurements of power consumption on sensor nodes that can be use to produce accurate models in a simulation environment. The sections that follow provide a brief overview of the sensor node and battery models and present the results of our power measurements.

2. SENSOR NETWORK AND NODE MODELS

A sensor network is modeled as a set of heterogeneous entities. Sensor nodes deployed over the area of interest are triggered by a certain set of stimuli that eventually result in a sensor report that is transmitted to a remote base station. Following this paradigm in a simulation environment, three main types of sensor nodes need to be created supported: 1) **target nodes** that stimulate the sensors, 2) **sensor nodes** to monitor events and 3) **user nodes** that query the sensors and are the final destination of the target reports.

The most interesting model is that of the sensor node. In addition to the communication protocol stack, this node model also includes a sensing stack that provides the interface to the sensor physical layer. The two stacks are connected together with an application layer, and together they constitute the algorithmic component of the node. To model power consumption, power models of the individual components are provided together with a battery model. As the protocols execute on the hardware, a corresponding amount of energy is depleted from the battery. Figure 1 provides an overview of the node model. This provides a flexible parametrizable model that can be applied to different sensor node architectures. The challenge in achieving an accurate sensor node model is to understand how the node consumes power.

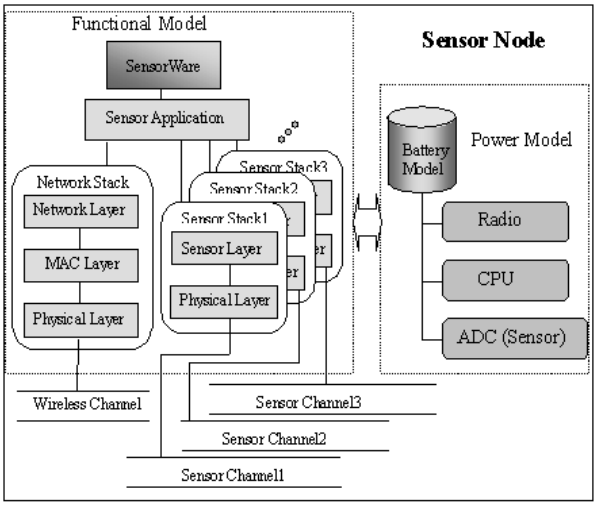


Figure 1 Sensor Node Model

3. BATTERY MODELS

3.1 Linear Battery Model

In this model, the battery is treated as linear bucket of energy. The maximum capacity of the battery is achieved regardless of what the discharge rate is. Such a model allows the user to examine the efficiency of applications by providing a simple metric of energy consumption. The remaining capacity after

Table 1 CPU Measurements for WINS and Experimental Nodes

CPU	Sensor	WINS		Experimental Node	
		Current	Power	Current	Power
Active	On	42.23mA	380mW	2.9mA	8.7mW
Sleep	On	7.0mA	64mW	1.9mA	5.9mW
Off	On	2.6mA	23.8mW	N/A	N/A
Power Down	Power Down	100μA	0.9mW	1μA	3μW

Table 2 WINS Radio Measurements

Radio Mode	Tx Power (mW)	Current Drawn (mA)	Power Consumed (mW)
Tx	0.12	43.64	395.99
Tx	0.16	43.68	396.35
Tx	0.23	43.82	397.61
Tx	0.30	43.95	398.77
Tx	0.44	44.14	400.48
Tx	0.95	45.5	412.68
Tx	1.32	45.95	416.72
Tx	1.78	45.86	424.87
Tx	2.51	47.77	433.03
Tx	3.47	48.68	441.18
Tx	10.00	59.59	538.63
Tx	13.80	63.23	571.02
Tx	19.05	68.23	615.43
Tx	27.54	73.68	663.70
Tx	36.31	79.14	711.94
Rx		41.41	375.96
Idle		38.68	351.4

Table 3 RFM Radio Measurements

Mode	Power Level	OOK Modulation				ASK Modulation			
		2.4Kbps		19.2Kbps		2.4Kbps		19.2Kbps	
	mW	mA	mW	mA	mW	mA	mW	mA	mW
Tx	0.7368	4.95	14.88	5.22	15.67	5.63	16.85	5.95	17.76
Tx	0.5506	4.63	13.96	4.86	14.62	5.27	15.80	5.63	16.85
Tx	0.3972	4.22	12.76	4.49	13.56	4.90	14.75	5.18	15.54
Tx	0.3307	4.04	12.23	4.36	13.16	4.77	14.35	5.04	15.15
Tx	0.2396	3.77	11.43	4.04	12.23	4.45	13.43	4.77	14.35
Tx	0.0979	3.13	9.54	3.40	10.35	3.81	11.56	4.08	12.36
Rx	-	4.13	12.50	4.13	12.50	4.13	12.50	4.13	12.50
Idle	-	4.08	12.36	4.08	12.36	4.08	12.36	4.08	12.36
Sleep	-	0.005	0.016	0.005	0.016	0.005	0.016	0.005	0.016

operation duration of time t_d can be expressed by following equation. Remaining capacity (in Amp*Hour) = $U = U' - \int_{t=t_0}^{t_0+t_d} I(t)dt$, where U' is the previous capacity and *I(t)* is the instantaneous current drawn from the sensor node at time *t*.

3.2 Discharge Rate Dependent Model

The maximum battery capacity is very much dependent on the discharge rate or the rate at which the current is withdrawn from the battery. At high discharge rates, the battery capacity is significantly reduced. To consider this effect of discharge rate dependency, we introduce factor *k* which is the battery capacity efficiency factor that is determined by the discharge rate. The definition of *k* is, $k = \frac{C_{eff}}{C_{tot}}$, where C_{eff} is the effective battery capacity and C_{tot} is the total rated capacity of the battery with both terms expressed in unit of Ampere*hour(Ah).

3.3 Relaxation Model

Real-life batteries exhibit a general phenomenon called "relaxation". The relaxation occurs when the discharge current from the battery is cutoff or reduced after draining the battery at high discharge rate. As the discharge rate of the battery drops, the battery's cell voltage recovers, and the battery has a chance to recover the capacity lost due to the high discharge rate. The relaxation phenomenon is adapted to our battery model to simulate the behavior of real life battery.

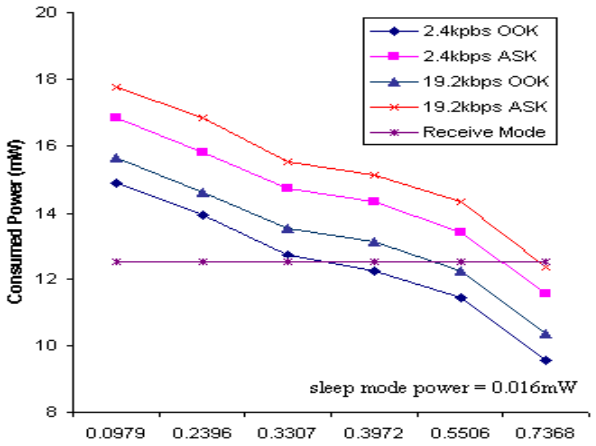


Figure 2 RFM Radio Power Comparisons

4. POWER CHARACTERIZATION

Sensor node power consumption depends on the node's mode of operation (receive, transmit, sleep, power down). During its lifetime, a node may switch between different operational modes according to a specific task or power management scheme. By measuring the power consumption at the different operational modes accurate models can be constructed and useful insight can be obtained about the individual components of the sensor nodes.

Using an HP 1660 oscilloscope and a high precision resistor we measured the power consumption of the radio and CPU of 2 types of nodes (WINS and Experimental nodes) at different operational modes. Table 1 shows the power measurements for the CPUs on the 2 nodes. The power consumption for the WINS radio is shown in table 2. The power consumption for the RFM radio is shown in table 2 and figure 2.

These measurements show some notable trends of how power consumption is distributed in a sensor node. In both nodes, the radio consumes the most power; 50-67% of the total power consumption. Furthermore, the difference in power consumption between transmission and reception in low power radios is very small. For the WINS radio the transmission power is at most 2 times greater than reception and 1.4 times greater for the RFM radio. At some power levels, the transmit power is smaller than the receive power (figure 2).

5. REFERENCES

[1] ns-2 simulator, http://www.isi.edu/nsnam/ns

[2] Wireless Integrated Networks Systems (WINS), http://wins.rsc.rockwell.com

[3] http://www.cs.berkeley.edu/~jhill/tos/

[4] DR3000 ASH Radio Module, http://www.rfm.com/products/data/dr3000.pdf

Appendix C:

Tasking Distributed Sensor Networks

Mark T. Jones Shashank Mehrotra
Jae H. Park

The Bradley Department of Electrical and Computer Engineering
Virginia Tech *

Abstract

Distributed wireless sensor networks provide significant new capabilities for automatically collecting and analyzing data from physical environments. Such sensor networks gather and analyze data in response to queries posed by users of the network. By using more sensor nodes that is strictly necessary to cover an area, these sensor networks can provide reliable information, tolerate many types of faults, and have a long effective lifetime. Like most wireless systems, however, these sensor networks must effectively manage power consumption to achieve a satisfactory system lifetime. Power management at the node level, including low power hardware and power aware operating systems, can provide significant savings in power consumption. In this paper, we present algorithms for managing power at the distributed system level, rather than just at the individual node level. These distributed algorithms determine which set of sensor nodes will be tasked with a given user query and which sensor nodes

*The authors gratefully acknowledge support for this work from the SenseIT program in the DARPA ITO office under contract F30602-99-1-0529.

will remain in an idle state conserving power. The goal of the algorithms is to extend the effective service time of the entire network. In this paper, we give a theoretical analysis of these algorithms along with extensive simulation results.

1 Introduction

A distributed sensor network is a system composed of many sensor nodes which communicate via a wireless network. Users query the distributed sensor network to acquire information in the area in which the network is operating. The sensor nodes gather information on their physical environment, such as acoustic data or seismic data, and then collaboratively process this data to form answers to user queries. These nodes are small and inexpensive to allow for the deployment of large numbers of nodes. By redundantly deploying more than the minimum number of sensor nodes required to cover an area, system reliability can be increased. Once deployed, it is difficult, if not impossible, to add a new energy supply to the nodes; they are limited to their initial energy supply plus what they can acquire from the environment. Power efficiency and conservation, therefore, are significant contributors to the lifetime of the network.

Power management must be incorporated at all levels of the sensor network design. Power management at the node level, including low power hardware and a power aware operating system, can provide significant savings in power consumption. This paper describes a method for assigning work (user queries) to the sensor nodes in such a way as to manage and balance power consumption among the sensor nodes. This query tasking process can

be managed by a central control point that allocates node-level tasks, oversees load balancing and network maintenance for the entire network. This solution, however, suffers the disadvantages posed by centralized systems, i.e. low fault tolerance, low scalability, communication bottlenecks, and the need for continuous updates. Instead, this paper presents distributed algorithms for managing power at the system level. These tasking algorithms determine which set of sensor nodes will be tasked with a given user query and which sensor nodes will remain in an idle state conserving power. Further, energy consumption is balanced among the nodes to avoid dead spots in the coverage area of sensor network.

This paper is organized as follows. Section 2 discusses ongoing research in distributed sensor networks. Section 3 presents the basic distributed algorithm for tasking the network. Section 4 explores variations on this tasking algorithm that a) prolong network life by efficiently utilizing the redundancy in node locations, b) load balance the network by selecting appropriate nodes to task for each query, and c) ensure good quality of sensor coverage for queries. Results from simulation models are presented in Section 5 that demonstrate that these distributed algorithms can increase the effective sensor network lifetime. Section 6 gives concluding remarks along with a discussion of future work.

2 Related Work

Distributed sensor networks are a broad and active area of research. Due to the unique operating requirements of these networks new approaches and designs are required at almost every level of operation. This section gives an overview of the work in this area that is

most relevant to this paper.

2.1 Node Level Architecture.

A primary goal in the design of sensor nodes is the creation of a very small, low power, inexpensive device. Work at UCLA and Rockwell has led to the integration of sensing, processing, and communication on micro-sensor platforms [16, 2]. The Smart-Dust project [8] explores an alternative to the traditional view of radio linked sensor networks. The project uses MEMS-based sensor nodes approximately a cubic millimeter in size that communicate with a base station or each other via a free space optical link. These nodes store no more than 1 Joule of energy [14] and exhibit power consumption at microwatt levels. The μAMPS project [11] is designing power-aware nodes, software for those nodes, and protocols between the nodes. A focus of this project is the interaction between software which can make energy-quality tradeoffs and hardware which can scale its energy consumption [12]. The PicoRadio project targets the energy efficiencies which can be realized through careful optimization of the radio network, particularly the physical, MAC, and network layers [17]. Rockwell Science Center has developed a micro-sensor unit with applications to target tracking, and environmental monitoring [10].

2.2 Network Layer Issues and Inter-Node Communication

Network layer design for large sensor networks [15] also has to work within the same energy conservation goals that motivate node architecture. Estrin, *et. al.* examine large sensor

networks and their applications in [3]. They propose a set of distributed algorithms running on each node that are responsible for all the sensing and communication tasks. A clustering algorithm for selecting local cluster heads is also described, and its advantages characterized in terms of robustness and energy savings. The same group discusses routing in sensor networks [22] and introduces two algorithms, the Basic Energy Conservation Algorithm (BECA) and the Adaptive Fidelity Energy Conservation Algorithm (AFECA). The former switches off the radio and trades off higher route latency for power savings. The latter uses information about node density to adaptively let neighboring nodes handle network traffic. Estrin, *et. al.* also describe a method to express communication patterns between the nodes via Directed Diffusion [5]. Pottie, *et. al.* [20] describe a protocol suite for sensor network elements beginning with the allocation and discovery of time slots to establishing routes among the nodes. A MAC protocol for interaction between a mobile node and the rest of the network is also discussed. These protocols try to minimize messages among nodes and the amount of information stored locally in registries to conserve energy as opposed to traditional protocols where energy constraints are not as strict. Iyengar, *et. al.* [6] propose a multi-level deBruijn network and show that this network achieves fault tolerance along with a a simple, decentralized routing scheme.

2.3 Energy Usage Characterization

Studies conducted by Katz and Stemm [21] provide information on the energy consumption patterns of small wireless devices/interfaces. They show that the time spent in the

idle state, where data is not actually being received or sent but the network interface is operational, dominates power consumption in these devices. Therefore, nodes that are not being tasked must conserve resources by switching off their network interface. Srivastava, *et. al.* have created SensorSim, a simulation infrastructure built upon models of energy consumption in sensor nodes. This infrastructure can be used for studying the basic aspects of a sensor network and can be combined with physical nodes to conduct hybrid simulations. By modeling data gathering and energy consumption in a sensor network, the authors in [1] establish upper bounds on the lifetime of a sensor network based on its energy consumption.

3 Election of Application Query Servers

This section introduces the concept of an Application Query Server (AQS). An AQS is similar in concept to a cluster head (c.f., [3]), but has more general responsibilities. An AQS is responsible for managing all of the sensor nodes in its geographic region of the network. This management includes assigning the tasks associated with user queries to the sensor nodes under its management; this task assignment should take into account the energy level of the nodes, the current workload of the nodes, and the capabilities of the nodes. The AQS advertises itself to the network routing layer to allow queries destined for its region to be routed to it. For example, in diffusion-based routing, the AQS would indicate a willingness to publish certain types of results. An AQS manages a set of sensor nodes in a region. The goal of the AQS is extend the effective lifetime of the network by

assigning tasks related to user queries to individual sensor nodes in a power-aware fashion. Achieving this goal requires that, as indicated in [21], nodes and their radios are shutdown (sleep mode) when their services are not required. In addition, energy consumption must be balanced across the nodes to avoid dead spots in the coverage of the sensor network and to allow nodes the opportunity to recover energy from the environment using, for example, solar cells. The responsibility for being an AQS must be rotated based on the energy levels of the nodes as well as faults that occur within a current AQS.

3.1 Application Query Server election: Basic Algorithm

This section describes a basic method for AQS selection that is fast, robust to failure, and power-aware. An analysis is given which proves that the algorithm scales well to large numbers of nodes Some definitions are required prior to presenting the algorithm.

S = set of sensor nodes

$|S|$= cardinality of S

s_i = sensor node i

$A(s_i)_j$ = coverage area of sensor node i for the sensor type j

$R(s_i)$ = radio coverage area of sensor node i

$D(S)_j$ = union of $A(s_i)_j$ for all i

node status = battery status, sensor capabilities, $A(s_i)$, $R(s_i)$, workload

G_r = the intersection graph where s_i is a vertex in the graph, and an edge e_{ij} exists if and only if $R(s_i)$ intersects with $R(s_j)$

A set of application query servers is chosen such that each s_i is either an application query server or $R(s_i)$ intersects with the $R(s_i)$ of an application query server. Note that these application query servers form a maximal independent set in the intersection graph G_r. Each application query server, s_k, maintains the node status of each s_j where $R(s_k)$ intersects with $R(s_j)$. The maximal independent set (MIS) in G_r is chosen by using a distributed, asynchronous algorithm shown in Figure 1. This basic algorithm is closely related to the coloring algorithm proposed in [7] which in turn draws on concepts from an algorithm in [9]. At each pass through the loop, the set of nodes, V, with random neighbors greater than their neighbors' random numbers is chosen from the remaining graph, H. This set, V, is added to the maximal independent set and then subtracted from H, along with all nodes which are neighbors of H. For the purposes of analysis and clear presentation, this algorithm is described as directly operating on graphs; in the actual implementation, no graphs are formed and the algorithm is operated on a purely distributed basis.

Given that the number neighbors of any particular node remains bounded as $|S|$ grows, this loop will execute $EO(\log(|S|)/\log\log(|S|))$ iterations [7]. In a typical sensor network, the number of neighbors should remain bounded as the size of a network increases; i.e., the node density remains constant as the number of sensor nodes is increased for more coverage. The expected running time grows *very* slowly with $|S|$, which indicates that the algorithm is scalable to large numbers of nodes.

The algorithm requires no global synchronization; all communication is purely nearest neighbor. At the beginning of the algorithm, each node broadcasts its random number to

1: assign each s_i a random number, $0 < r(i) < 1$
2: $P = S$
3: $I = \emptyset$
4: $H = G_r$
5: while $(P \neq \emptyset)$ do
6: $V = \{s_i \in H : r(i) > r(j)\ for\ each\ e_{ij}\}$
7: $P = P \setminus V$
8: $P = P \setminus$ adjoining (V)
9: $I = I \cup V$
10: H is now the graph induced by P
11: end while
12: at this point, I forms the MIS

Figure 1: The basic algorithm for AQS selection.

its neighbors (anyone who can hear). After that, the only information exchanged between nodes is an indication of when a node is included, or not included, in the independent set; this information is sent by a node only to its nearest neighbors whose random number is lower than its random number (though broadcast is also acceptable). The asynchronous nature of the algorithm is illustrated in Figure 2, where node seven must wait on information from nodes one and five before making its decision on inclusion in the independent set. Once node seven has reached its decision, it broadcasts its status to its remaining neighbors who have been waiting because their random numbers are lower than node seven's random number.

The algorithm, as presented, assumes symmetric and reliable communication between pairs of nodes. The effect of lost messages on the algorithm is now considered. Consider a failure in the exchange of r values between s_i and s_j such that s_i receives $r(j)$, but s_j does

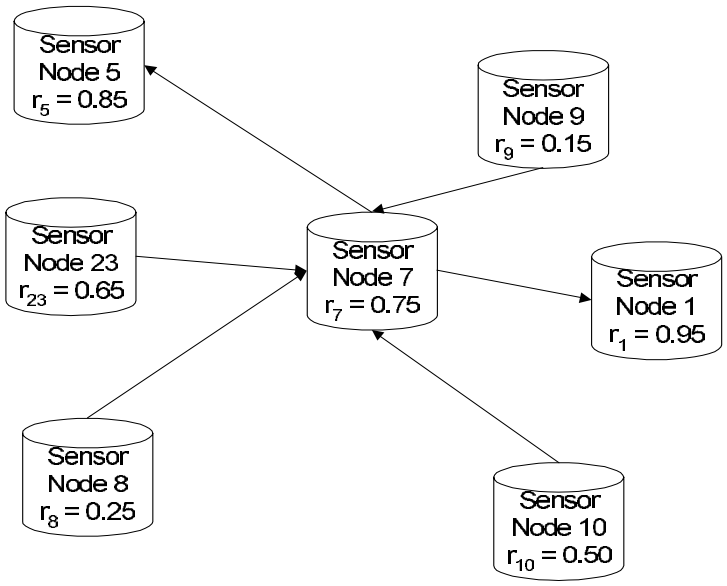


Figure 2: An example scenario for an AQS election. The direction of information flow from various nodes with respect to node seven is shown.

not receive $r(i)$ (unsymmetric communication). If $r(i) > r(j)$, then both nodes will proceed normally through the algorithm, with the potential that both of them may become an AQS. An extra elected AQS can be considered inefficient, but if the communication between the two nodes is unreliable then this may be a desirable result. If $r(i) < r(j)$, then s_j will proceed normally and correctly through the algorithm and s_i will wait for a message from s_j indicating whether or not s_j is an AQS. If the message is received (assuming that s_j is broadcasting to neighbors), then s_i proceeds normally and correctly through the algorithm. If this message is not received (assuming that s_j is communicating directly to its neighbors and does not know about s_i), then s_i will wait indefinitely for a message. After a timeout period, if for any reason a node does not receive a message it expects from a neighbor regarding inclusion in the independent set *and* none of its neighbors have been elected as

an AQS, then it elects itself as an AQS and lets its neighbors know. Again, this may result in an extra AQS, but if communication is unreliable then this may be a desirable result.

3.2 Algorithm Variants to Improve Resource Utilization

The basic algorithm is not influenced by the node status. For example, it might be desirable to choose Application Query Servers from those nodes with the highest battery life. In that case, one could define remaining battery life on an integer scale of 0 to k. The above algorithm can be modified to redefine $r(i) = battery\ life + a$, where a is random number between 0 and 1. With this modification, the nodes with the longest remaining battery life will always be elected, but ties will be broken randomly. The communication patterns are unchanged. This technique can be expanded to include any function of desired node characteristics. Related work in [4] shows that the expected running time of this algorithm remains $EO(\log(|S|)/\log\log(|S|))$.

The algorithm can also be extended to different types of intersection graphs beyond the simple radio connectivity graph. For example, one may want to select an MIS based on the sensor range rather than the radio range. In this case, the intersection graph upon which the MIS is chosen is different, but the algorithm remains the same. If the sensor range is less than or equal to the radio range, then the required radio communication is still only a single hop. If the sensor range is greater than the radio range, then the communication is no longer a single hop, but the algorithm is unchanged.

4 Tasking the Network with AQSs

This section considers the operation of the network following election of AQSs. User queries are sent to relevant AQSs and it is the responsibility of an AQS to assign tasks related to those queries to nodes within its responsibility. To accomplish this task, an AQS must maintain a record of the status of nodes within its area of responsibility. This status includes the remaining battery power on the node, the current state of the node (awake, asleep, and dead), the sensor capability of the nodes, and the current tasks assigned to the node. This requires nodes to periodically inform neighboring AQSs of their battery status. The remainder of this section describes how an AQS uses status information to assign tasks to nodes within its region of responsibility.

4.1 Determination of Responsibility

An AQS is responsible for the physical area associated with the sensors of all of its neighboring nodes. Note that a node could have more than one neighboring AQS; in this case, it In addition, a query could be sent to a region that has multiple AQSs; i.e., the area in question extends beyond the region associated with a single AQS. Methods for assigning tasks must take both of these overlapping situations into account to avoid overloading a node.

The basic problem is that one node could be assigned tasks by two different AQSs simultaneously; in this assignment, neither AQS knows the other is making an assignment and, therefore, they may overload that node. To resolve this problem without extensive

coordination between AQSs, a node notifies each neighboring AQS whenever it is assigned a task (this can be accomplished via broadcast). Each neighboring AQS then acks that message. If a node receives a second task assignment from any AQS before it receives an ack from that AQS, then it assumes the assignment was made without considering its current load and it rejects the assignment. The AQS must then decide to reaffirm the assignment or find a substitute. This method of conflict resolution allows all boundary conflicts to be avoided while still requiring only nearest neighbor communication.

4.2 Energy Efficiency and Task Assignment

As noted before, the high degree of redundant nodes that is a feature of these networks allows for only a subset of nodes in a given region to be tasked to service any individual query. When not tasked, a node must enter a low-power state to conserve energy; to be effective this sleep state requires that the radio interface be powered down [21]. In this case, the AQS could determine the sleep period and then wait for the node to power up before re-tasking it. This presents a difficulty if tasks arrive and that node's resources are required before it is awake again. A low-power wakeup radio, similar to a pager, can be attached to a node which allows for the node to be woken up on demand; this wakeup radio allows for little, if any, information to be sent, its function is solely to wake the node up [19]. This wakeup radio can significantly improve performance and its use is assumed in the simulations; it is not, however, required for proper functioning of these algorithms. Nodes transitions between several states as shown in Figure 3. The *awake/idle* state is where the

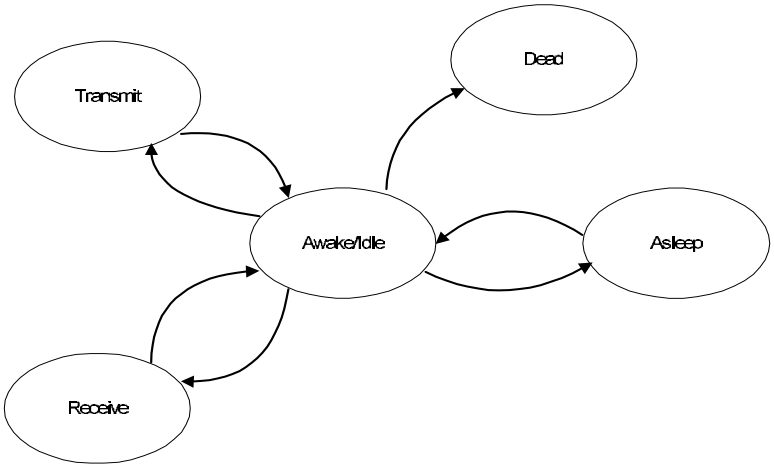


Figure 3: Transitions between states at a node.

network interface is operational and the node can send or receive data. The *asleep* state has already been described. A node enters the *transmit* or *receive* states to send or receive data, and returns immediately to the *awake* state. The *dead* state is reached when the battery of the node is completely drained out, and it cannot function anymore.

An AQS may use two approaches to process a user level query into task assignments for nodes. A simplistic approach, *Approach I*, involves receiving a query at the AQS, and activating all nodes in its range to retrieve the information required by the query. Approach I requires relatively little information to be maintained at the AQS, and in some sense is a brute force approach to tasking. The second approach, *Approach II*, is motivated by the fact that nodes need to be kept asleep as long as possible to achieve meaningful energy savings. Therefore, Approach II tries to select a set of appropriate nodes to task by taking into account the remaining battery life at each node, number of active tasks at a node, and the geographical region of the query. All other nodes in the region of the AQS are allowed

to transition to the asleep state.

4.2.1 The Grid Map

In Approach II, an AQS forms a *grid map* by subdividing its region into a rectangular grid. The purpose of this map is to represent sensor coverage of various nodes in the AQS's region. Therefore, the size of the grid map must be such that it includes the sensor coverage of all the nodes that may lie in its region of responsibility.

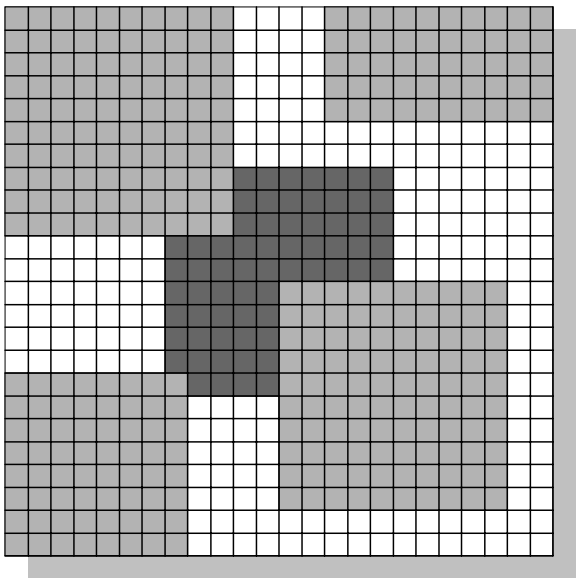


Figure 4: Representation of a grid map at an AQS.

A cell in the grid map is colored to denote the relative position of a node from the AQS. Cells around that particular cell are colored with the same color to indicate the sensor coverage of that node. Eventually, when such a map has been constructed, a colored cell on the grid map represents the presence of sensor coverage at that location, while

an un-colored cell denotes the absence of sensor coverage. Such a grid map is shown in Figure 4. The center of this map is drawn relative to the position of the AQS. The black region corresponds to sensor coverage provided by the AQS itself. Other shaded regions correspond to nodes around the AQS within its region. All un-shaded blocks correspond to uncovered area. The assignment of tasks to nodes is a two step process. The first step consists of constructing the grid map by selecting nodes that satisfy some fitness criteria. The second step consists of actually assigning tasks to a subset of the nodes that are currently in the grid map.

4.3 Construction of the Grid Map

At all times during its operation, the AQS maintains a set B that is a subset of all the nodes in its region. This set essentially contains those nodes that *can* be used to service a query that is sent to its region. The nodes in B are placed on the grid map and the sensor coverage associated with them is used to color the pixels of the grid map.

Before discussing how the set B is chosen, the following nomenclature is defined. Define set $N = \{s_1, s_2 \ldots s_n\}$ to be all the *alive* nodes in the region of the AQS. Further, define $b(s_i)$ to be the current energy level of node $s_i \in N$. Finally, define r_{index} to the redundancy index, i.e. the value of the sensor coverage overlap between two nodes at which they are considered to be redundant with respect to each other.

The selection of B is described using the algorithm given in Figure 5. The operation of the algorithm can be explained as follows. In steps 3 and 4, a node that does not have any

coverage overlap with any node currently in the grid map is added to set B. If a node s_i does have an overlap with a node $s_j \in B$, two cases may occur. If the overlap of s_i and s_j exceeds the value of the redundancy index r_{index}, then the node with the greater energy level is selected in B and the other node is removed from B (steps 8, 9 and 10). Alternatively, if the two nodes are not redundant both are added to B (step 13). To summarize, a node is added to set B under 3 conditions, (a) if it has no overlap with any node currently on the grid map, (b) if it is redundant with a node in the grid map with a lower energy level than itself, and (c) if its coverage overlaps but is not redundant with any node on the grid map. The only condition under which a node is removed from the grid map is when it is redundant with another node with a greater energy level.

```
1:      B = ∅
2:      for all s_i ∈ N do
3:          if (A(s_i)_k ∩ A(s_j)_k) = ∅  ∀ s_j ∈ B then do
4:              B = B ∪ s_i
5:          else
6:              for all s_j ∈ B : A(s_i)_k ∩ A(s_j)_k) ≠ ∅ do
7:                  if (A(s_i)_k ∩ A(s_j)_k) > r_index then do
8:                      if b(s_i) > b(s_j) then do
9:                          B = B ∪ s_i
10:                         B = B \ s_j
11:                     end if
12:                 else
13:                     B = B ∪ s_i
14:                 end if
15:             end do
16:         end if
17:     end do
```

Figure 5: Algorithm for constructing the grid map.

4.3.1 Using the Grid Map to Assign Tasks

Following the construction of the grid map, an AQS attempts to assign tasks to the nodes in the grid map (set B) by using the algorithm described in Figure 6. Prior to a discussion of this tasking algorithm some additional terminology is introduced. Define A to be the set of all the currently *awake* nodes that are being tasked, and S to be the set of all the *asleep* nodes. Therefore, $N = A \cup S$. Q is the set of currently active queries at an AQS. Each query $q \in Q$ is organized as the following tuple: $\{q_{id}, \{s_1, s_2 \ldots s_m\}\}$ where q_{id} is the query identifier and $s_1, s_2 \ldots s_m$ are the nodes that are being tasked by this query.

It is observed that many queries sent to an AQS in the network may require the AQS to task only a portion of the region it covers. Therefore, not all nodes in set B need to be in the *awake* state. Only a subset of the nodes in B, i.e., A need to be *awake* at a given time to actually service those queries. The remainder of the nodes are in the *asleep* state, i.e. S. The algorithm in Figure 6 describes the partitioning of nodes in N into sets A and S.

The important step in this algorithm is the method for the selection of the query tuple for a query q in step 5. This is done by calculating the intersection of the geographical area specified by the query with the corresponding area in the grid map. All nodes whose sensor coverage lies within this region are added to the tuple. In step 9, the AQS calculates if a node in B needs to be *awake* for any of these queries. All such nodes are added to the set A. Once the construction of these sets is complete, nodes in A are sent wakeup messages and task assignments whereas the nodes in S are sent messages that enable them to enter

1: $S = \emptyset$
2: $A = \emptyset$
3: for each $q \in Q$
4: $q = \{q_{id}, \{\emptyset\}\}$
5: select $\{s_1, s_2 \ldots s_m\}$ in B which will service q
6: $q = \{q_{id}, \{s_1, s_2 \ldots s_m\}\}$
7: end for
8: for each node $s_i \in B$
9: if $\exists\ q \in Q : q$ is serviced by s_i, and $s_i \notin A$
10: $A = A \cup s_i$
11: end for
12: $S = N \setminus A$

Figure 6: Algorithm for assigning tasks to nodes in the grid map.

the *asleep* state.

Because nodes in A report their battery level to the AQS periodically, the grid map must be redrawn to reflect these updated values. Each time the grid map is redrawn, sets B and A are re-calculated and new nodes take over the existing queries. To understand the effect of reconstructing this grid map, consider the following scenario. Battery updates are received very often from nodes that are awake and the period for redrawing the grid map is small. In this situation, nodes are quickly toggled between the awake and asleep state in the grid map. This leads to a situation where the set B, and therefore the set A are re-calculated too rapidly. On the other hand, if the battery updates are received at widely spaced intervals and the grid map redraw periods are comparatively large, sets B and A remain fairly constant for long periods of time. This may lead to the excessive energy consumption of the nodes in A. Therefore, the frequency of these updates and the

frequency of reconstructing the grid map should be infrequent when the activity in the network is low, and somewhat higher for more active periods of the network.

4.4 Quality of Sensor Coverage for User Queries

While extending the life of the network by increasing $|S|$ is an important objective of these tasking models, it is also necessary to ensure that adequate sensor coverage is achieved for all queries. Because there are at most $|N|$ nodes which can be tasked by the AQS, we cannot achieve coverage greater than would be achieved by activating all those nodes simultaneously. Two factors determine whether we can achieve the quality of coverage equivalent to turning on all nodes.

(a) The first is the criteria used to determine the value of the redundancy index r_{index}. It is obvious that if a node's sensor coverage has no overlap with any other in node in N, then it must be included in B. Also, a node s_i may be replaced in the grid map by node s_j if s_i and s_j are located at the same position, and $b(s_j) > b(s_i)$. To conserve resources, the restriction that the nodes be exactly co-located can be relaxed; for example two nodes may be defined to be replaceable with each other if the coverage of one overlaps that of the other by 80%. This percentage, i.e., the redundancy index (RI), is varied in simulation experiments. The drawback of this approach is that the queried area and the actual tasked area may differ if low values are used for the RI. This heuristic however, provides a means to tradeoff network life with the quality of coverage. It may also be employed to increase the life of the network for periods when queries have relatively non-

critical boundary requirements, or when prolonging the network life is more important than maintaining high fidelity for queries.

(b) The second factor that has bearing on sensor coverage is how nodes are added to the query tuples in step 5 of Figure 6. This method involves intersecting the area defined by the query with the corresponding region in the grid map and selecting nodes that lie in the intersection. The shape of the query and the approximation of a sensor's coverage in the grid map must accurately represent the region of the query and the actual sensor coverage respectively. For this implementation a simple rectangular representation is chosen for both the queries and the sensor coverage. This simplifies the calculation of the intersection. These representations can be extended to better approximations. Note that for queries that lie across multiple AQSs, the sum of queried areas at those AQSs provides the total coverage.

Note that some collaborative signal processing algorithms may require the use of overlapping sensor node coverage areas; the methods for choosing B and adding nodes to query tuples can be modified to address this requirement.

5 Simulation Experiments

5.1 Simulation models

Simulation models based on the above algorithms were constructed using Opnet Modeler v6.0 [13]. A finite state machine is used to represent the function of a node in the network.

This finite state machine has two distinct parts. The first part contains logic for the election of AQSs using the Maximal Independent Set algorithm. The second part is based on the transition diagram shown in Figure 3; this part contains logic for nodes to perform sensing tasks, transmit/receive packets, or sleep until woken up. The sensor node energy consumption models are based on the work of Stemm and Katz [21], and Srivatava, *et al.* [18]. At the start of the simulation each node begins with a fixed energy quota of 1 Joule. The energy depleted in the various states computed according to the following ratios (*asleep : awake : receive : transmit*) $(0 : 1 : 1.034 : 1.531)$. Define T_{bl} to be the lifetime of an *awake* node that does not engage in any sensing or communication task; this value is calculated to be one day. Each simulation is run for one week.

Additional energy is consumed by a node if it is engaged in sensing tasks; this extra energy is modeled as a linearly increasing function of the number of currently active tasks. A node is assumed to have reached the *dead* state when it's energy falls below 10% of the original. Nodes are equipped with a radio transmitter/receiver module supporting a bit-rate of 2.4 Kbps over a 25m range. Each node also has a seismic sensor capable of detecting activity within a radius of 10m.

5.2 Simulation Scenarios

The following two scenarios were constructed to evaluate performance criteria pertaining to energy, effective coverage, and network lifetime. A scenario has a set of parameters that define the purpose of the network, the positions where nodes are placed, and the queries

posed to the network.

5.2.1 The Scenarios

In Scenario I, a set of sensor nodes is placed in a rectangular region for the purpose of providing situational awareness to users traversing the region. In this scenario, users walk through the region along a straight line at a constant speed. The user generates queries for the area immediately surrounding him or her such that the area around the person is always tasked to return information about the environment. The nodes are placed in a 2-D hexagonal mesh structure such that inter-node separation is constant; this configuration is optimal for this scenario.

In Scenario II, the sensor network is employed to detect intrusions across a fixed perimeter. The sensor network is always tasked with the responsibility of detecting intrusions across the perimeter. In this case, the nodes are evenly distributed along the perimeter in an optimal fashion as shown in Figure 7. Scenario II differs from scenario I in the volume and nature of queries; it attempts to load the network with large, continuously operating queries at fixed locations.

5.3 Metrics for Energy Usage and Quality of Coverage

For both scenarios, simulations were performed using Approach I and Approach II. To understand the patterns of energy consumption for these approaches, the following statistics were defined and collected.

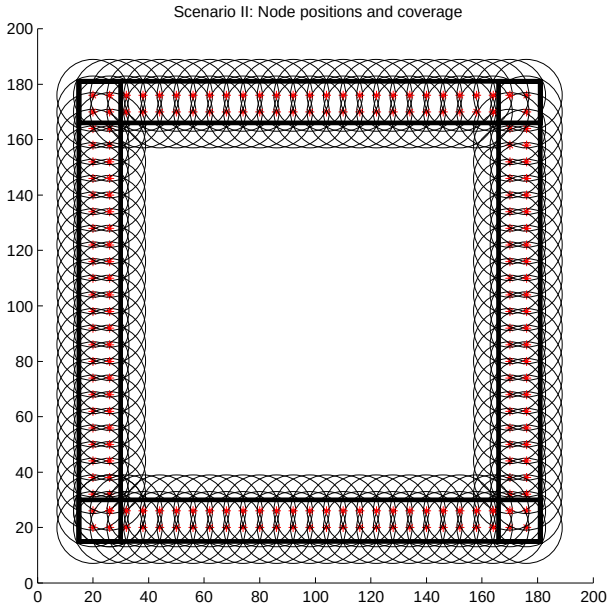


Figure 7: A circle represents the coverage radius of a sensor. The rectangles represent the four queries along the sides of the perimeter

Metric 1 *Average energy value of all the nodes in the network.* This measure is a simple expression of the total energy content of the network at a given time. When plotted against time it gives an indication of the lifetime of the network.

Metric 2 *Minimum energy of all the nodes in the network.* One of the attempts of our tasking algorithm is to balance power consumption across all nodes. The minimum energy value indicates how well that objective is achieved.

Metric 3 *Total energy consumed in the awake, transmit and receive states for each node.* The breakup of these values provides insight into the modes of operation of individual nodes.

Metric 4 *Maximum time spent by any node in electing a new AQS.* As the network size increases, the scalability and running time of this algorithm is tested by calculating how much time is spent in choosing a new AQS.

Along with extending the lifetime of the network, it is important to ensure that the network is delivering on its overall objective, i.e. good sensor coverage throughout the lifetime of the network. Two measures are defined to gauge the current coverage in the network.

Metric 5 *% of uncovered area in the region.* This metric is defined as the percentage of area over the entire region not covered by any sensor at a given time.

Metric 6 *Time at which the first breach occurs.* For Scenario II, an obvious quality measure is how long the integrity of the perimeter being sensed is maintained. When enough nodes have died so as to enable an entity to cross the perimeter without detection, a breach occurs and the sensor network effectively fails.

5.4 Results: Scenario I

5.4.1 Experiments with Constant Node Density

Tables 1 and 2 list the parameters that define simulation sets #1 and #2 respectively. In Set #1, different numbers of nodes are placed in a regular 2-D hexagonal grid structure. These nodes are placed such that the node density or the inter-node separation is constant. In contrast, in Set #2 different numbers of nodes are placed randomly in a rectangular area. Because nodes are placed randomly for Set #2, the node density is defined as the

ratio of the number of nodes and the total area in which these nodes are placed. For each Simulation ID in a set, simulations are conducted using Approach I and Approach II. As explained earlier in Section 4, Approach I is the simplistic tasking approach that is used to benchmark the distributed tasking approach, i.e., Approach II.

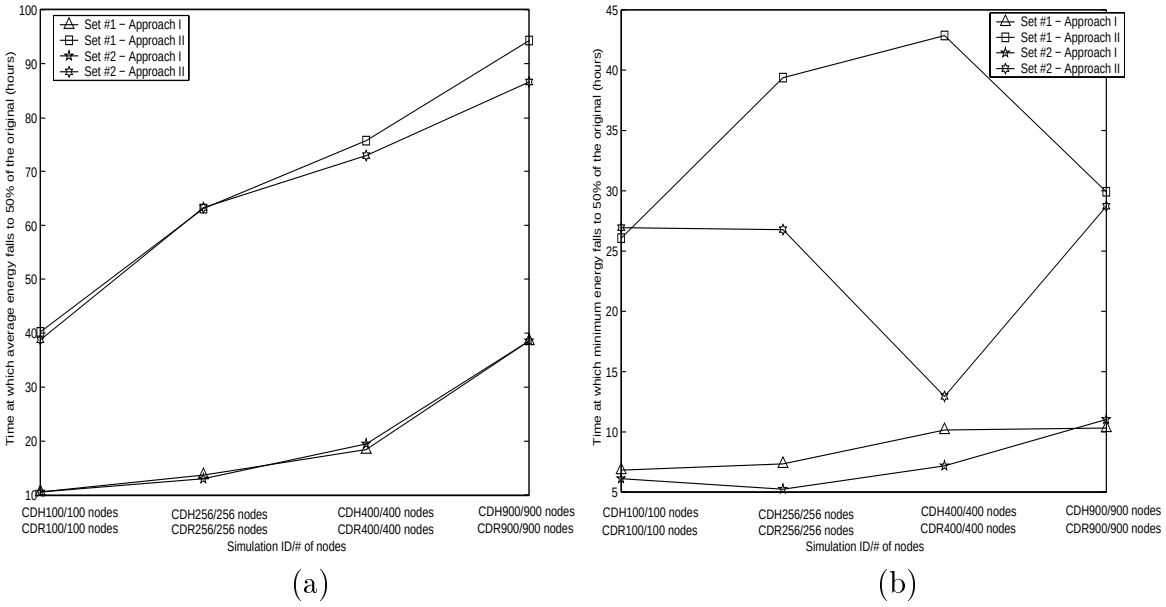
Table 1: Set #1 (Scenario I, **C**onstant Node **D**ensity, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*	Area
CDH100	100	0.90	100m x 100m
CDH256	256	0.90	160m x 150m
CDH400	400	0.90	200m x 185m
CDH900	900	0.90	300m x 275m

Table 2: Set #2 (Scenario I, **C**onstant Node **D**ensity, **R**andom Node Placements)

Sim ID	# of Nodes	*RI*	Area
CDR100	100	0.90	100m x 100m
CDR256	256	0.90	160m x 150m
CDR400	400	0.90	200m x 185m
CDR900	900	0.90	300m x 275m

Results for Metric 1 (average energy) and Metric 2 (minimum energy) are shown in Figure 8(a) and Figure 8(b) respectively. The metrics are plotted as follows; the time taken in hours for the average/minimum energy to fall to 50% of the original is plotted on the Y axis vs. the Simulation ID that is plotted on the X axis. From Figure 8(a) it can be seen that Approach II shows a marked improvement over Approach I in extending the lifetime of the network. Similarly, the minimum energy plot shown in Figure 8(b) indicates



(a) (b)

Figure 8: (a): Time taken for the average energy of all nodes to fall to 50% of it's original value. (b): Time taken for the minimum energy of all the nodes to fall to 50% of it's original value.

the extent to which battery of *weakest* node in the network is protected. This is also shown to improve for Approach II over Approach I.

From these figures it is observed that a comparison of Set #1 and Set #2 with respect to Metric 1 indicates very similar results. However, the hexagonal placement (Set #1) yields better results if we consider Metric 2. This leads to the conclusion that whereas the average energy is largely unaffected by the random placement (as opposed to the hexagonal placement), the effect on the minimum energy is distinctly visible. Notwithstanding this observation, observe that the minimum energy statistic for the random placement does not show a well defined trend. This may be due to areas in the network that are covered only by a single node. If this node is tasked excessively its energy is depleted rapidly. In contrast, if it is not tasked as often the minimum energy statistic may actually improve as can be seen from Figure 8(b). Therefore, it may not be possible to adequately predict the minimum energy value for different random node placements. This a potential drawback for a random placement because the minimum energy value represents the first failure in the network.

5.4.2 Experiments with Varying Node Density

Tables 3 and 4 list the parameters that define simulation sets #3 and #4 respectively. In Set #3, a fixed number of nodes is placed in different size areas. In contrast, Set #4 contains a group of simulations where different numbers of nodes are placed in a fixed area size. The purpose of these experiments is to evaluate the performance of the system as a

function of the node density.

Table 3: Set #3 (Scenario I, **V**arying Node **D**ensity, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*	Area
VDH256a	256	0.90	100m x 100m
VDH256b	256	0.90	160m x 150m
VDH256c	256	0.90	200m x 185m
VDH256d	256	0.90	300m x 275m

Table 4: Set #4 (Scenario I, **V**arying Node **D**ensity, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*	Area
VDH100e	100	0.90	100m x 100m
VDH256f	256	0.90	100m x 100m
VDH400g	400	0.90	100m x 100m
VDH900h	900	0.90	100m x 100m

Figure 9(a) shows the performance of both sets with respect to Metric 1. For Set #3, consider the average energy plots for Approach II and Approach I. As the distribution of nodes in an area becomes more sparse, the performance of Approach II begins to approach that of Approach I. In contrast, the average energy plots for Set #4 indicate that an increase in node density leads to an appreciable increase in the life of the network. Figure 9(b) shows the performance of both sets with respect to Metric 2. The trend observed is nearly identical to that for Metric 1. The results obtained from these experiments with node density are expected because Approach II inherently relies on node redundancy to achieve energy savings.

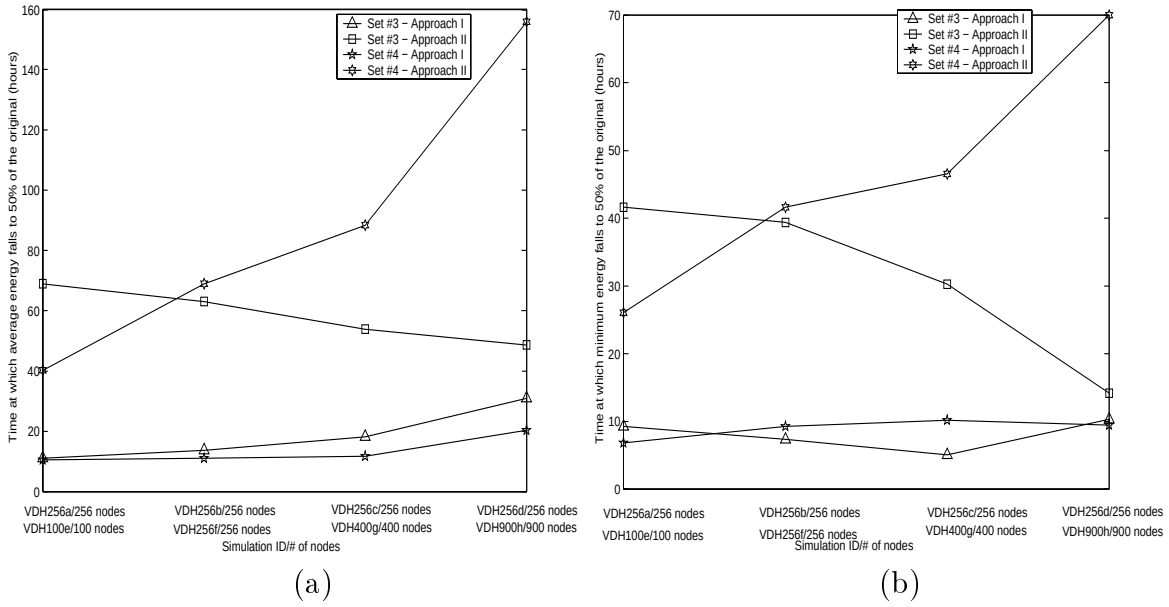


Figure 9: (a): Time taken for the average energy of all nodes to fall to 50% of it's original value. (b): Time taken for the minimum energy of all the nodes to fall to 50% of it's original value.

5.4.3 Energy Consumption for a Single Node

Table 5 displays the results for the individual energy consumption of a node in the *awake*, *transmit*, and *receive* states for Set #1. Only a small fraction of the total energy is consumed in communication with the balance being consumed in the awake state. Observe that nodes spend more time receiving messages in Approach I than in Approach II. This is an expected result, primarily because in Approach I all nodes in a region are woken up for each task thereby causing them to listen over the radio channel for longer periods.

Table 5: Average % energy consumed in the *awake*, *transmit*, and *receive* states

Set #1: Approach I				Set #1: Approach II			
Sim ID	*awake*	*transmit*	*receive*	Sim ID	*awake*	*transmit*	*receive*
CDH100	99.939	0.013	0.048	CDH100	99.506	0.081	0.413
CDH256	98.765	0.039	1.196	CDH256	99.437	0.087	0.477
CDH400	98.806	0.039	1.155	CDH400	99.416	0.086	0.499
CDH100	98.803	0.037	1.161	CDH900	99.404	0.082	0.515

5.4.4 Experiments with Varying Redundancy Index

Tables 6 and 7 list the parameters that define simulation sets #5 and #6 respectively. The effect of varying the redundancy index (RI) is studied in these sets. In Set #5, nodes are placed in a hexagonal formation; the density and area of the network are constants. In Set #6, nodes are placed randomly; the density and area of the network are constants. For both sets the only variable parameter is the RI.

Figure 10(a) shows the performance of both sets with respect to Metric 1. The life of

Table 6: Set #5 (Scenario I, **V**arying **R**edundancy Index, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*	Area
VRH256a	256	0.90	100m x 100m
VRH256b	256	0.70	100m x 100m
VRH256c	256	0.50	100m x 100m
VRH256d	256	0.30	100m x 100m

Table 7: Set #6 (Scenario I, **V**arying **R**edundancy Index, **R**andom Node Placements)

Sim ID	# of Nodes	*RI*	Area
VRR256a	256	0.90	100m x 100m
VRR256b	256	0.70	100m x 100m
VRR256c	256	0.50	100m x 100m
VRR256d	256	0.30	100m x 100m

the network does not show any appreciable connection with the RI for either the hexagonal (Set #5) or the random (Set #6) placement of nodes. Likewise, the results for Metric 2 shown in Figure 10(b) also do not display any significant trend. However, the tasking algorithm is shown to be effective across different values of RI; Approach II outperforms Approach I across all these values.

5.4.5 Scalability of the AQS Election Algorithm

The time taken by nodes to elect AQSs is collected to demonstrate that the execution time of the algorithm is scalable as the number of nodes increases as well as to show the effect of node density on the execution time. Figure 11 shows the times taken to elect AQS for Set

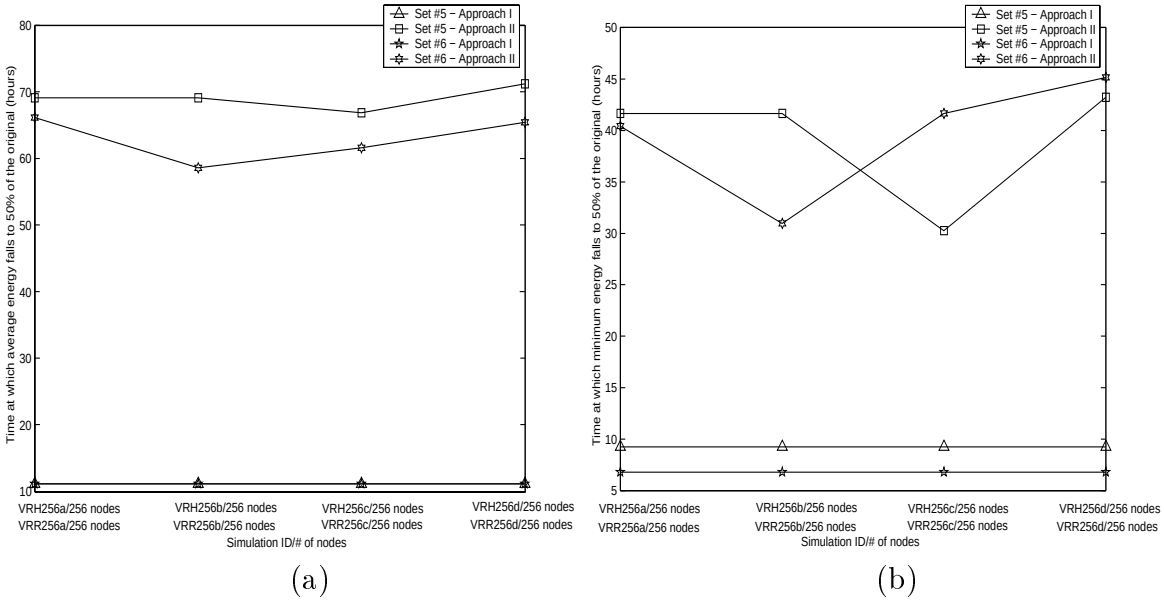


Figure 10: (a): Time taken for the average energy of all nodes to fall to 50% of it's original value. (b): Time taken for the minimum energy of all the nodes to fall to 50% of it's original value.

#1 and Set #4. For Set #1, the election time remains nearly constant as the number of nodes increases. This is because the density of nodes is constant. The effect of increasing the node density (Set #4) is to increase the time taken to elect AQSs. This is because the running time of the algorithm is affected by the number of neighbors for each node.

5.4.6 Experiments to Determine Quality of Sensor Coverage

Statistics were collected to compare the effective coverage resulting from the user of each algorithm. Figure 12(a) shows a plot of the time at which the first uncovered area is recorded for Set #1 using Approach II and Approach I. Figure 12(b) depicts a plot of the % uncovered area vs. time for 256 and 900 nodes. This displays the coverage characteristic

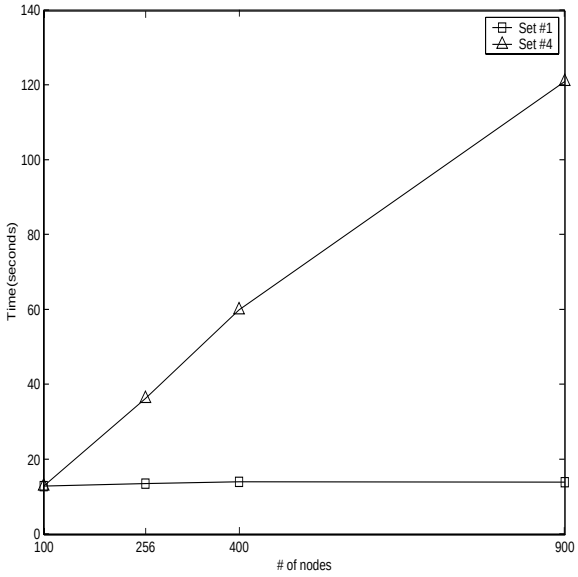


Figure 11: Time taken to elect an AQS for an increasing number of nodes.

over the entire simulated time. From both these figures, observe that Approach II is able to improve coverage by several orders of magnitude throughout the lifetime of the network (note that both algorithms start with 0% uncovered and must eventually end with 100% uncovered).

5.5 Scenario II

5.5.1 Experiments with Varying Redundancy Index

Table 8 lists the parameters that define Set #15. For this set, nodes are placed around a rectangular perimeter for the perimeter monitoring application. The energy consumption behavior for this scenario is analyzed using Metric 1 (average energy) and Metric 2 (minimum energy). Experiments conducted for this scenario assume that the placement and

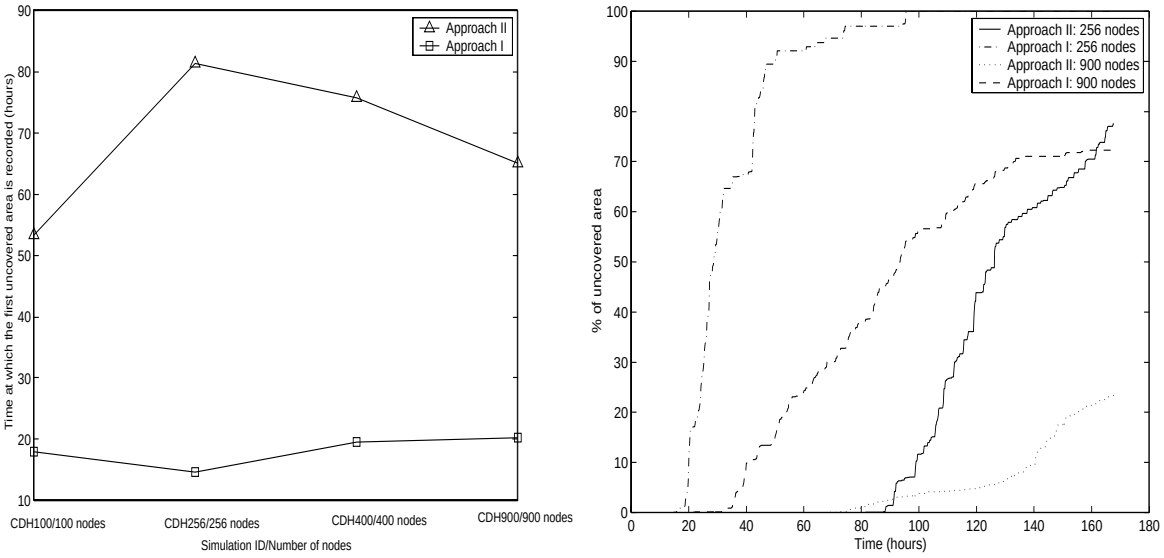


Figure 12: (a): Time at which the first uncovered area is recorded vs. Simulation ID. (b): Plots of % uncovered area vs. Time for different number of nodes.

density of nodes around the perimeter is fixed. The value of the RI is varied to gauge its effect on these metrics.

Table 8: Set #7 (Scenario II, **V**arying **R**edundancy Index, **P**erimeter Node Placements)

Sim ID	# of Nodes	RI	Area
VRP200a	200	0.90	200m x 200m
VRP200b	200	0.70	200m x 200m
VRP200c	200	0.50	200m x 200m
VRP200d	200	0.30	200m x 200m

Similarly, the energy consumption behavior for Scenario II was investigated. The average and minimum energy statistics for Set #7 is given in Figure 13. Set #7 explores the energy characteristics for Scenario II. While decreasing the RI impacts the lifetime of

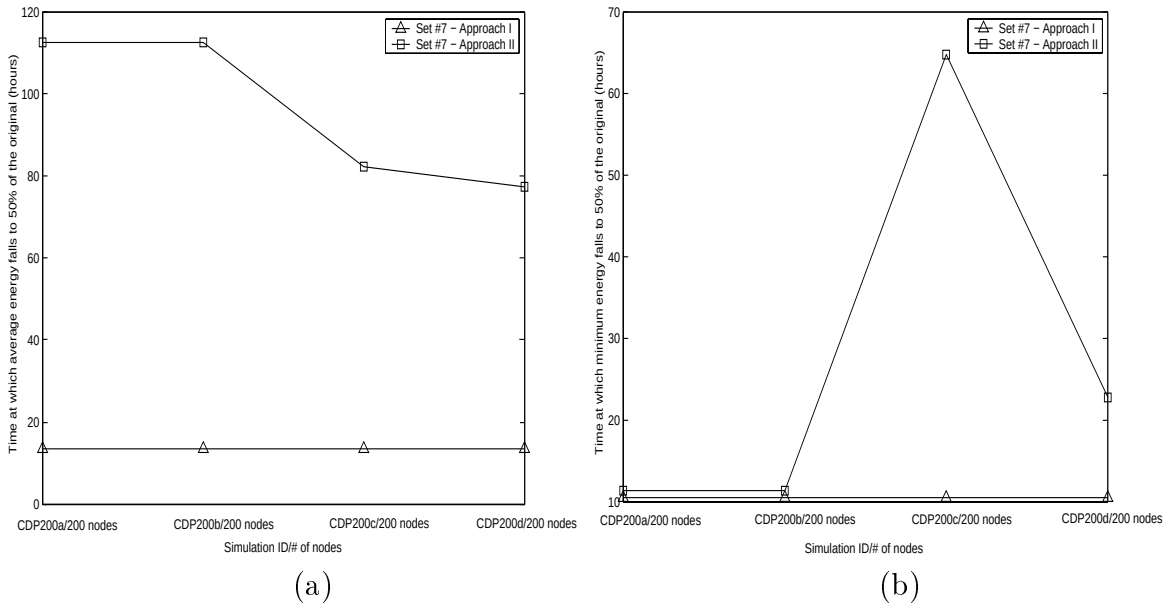


Figure 13: (a): Time taken for the average energy of all nodes to fall to 50% of it's original value. (b): Time taken for the minimum energy of all the nodes to fall to 50% of it's original value.

the network negatively, the minimum energy characteristic shows some improvement with lower values of *RI*.

5.5.2 Experiments to Determine Quality of Sensor Coverage

Figure 12(a) shows a plot of the time at which the first uncovered area is recorded for Set #1 using Approach II and Approach I. Figure 12(b) depicts a plot of the % uncovered area vs. time for *RI* values of 0.70 and 0.30 respectively. Though the first uncovered area is recorded at a later time for lower values of *RI*, the overall coverage deteriorates much more rapidly than for higher values of *RI*. Figure 15 shows the times at which the first breach across the perimeter was recorded. For lower values of *RI*, a breach is observed earlier in

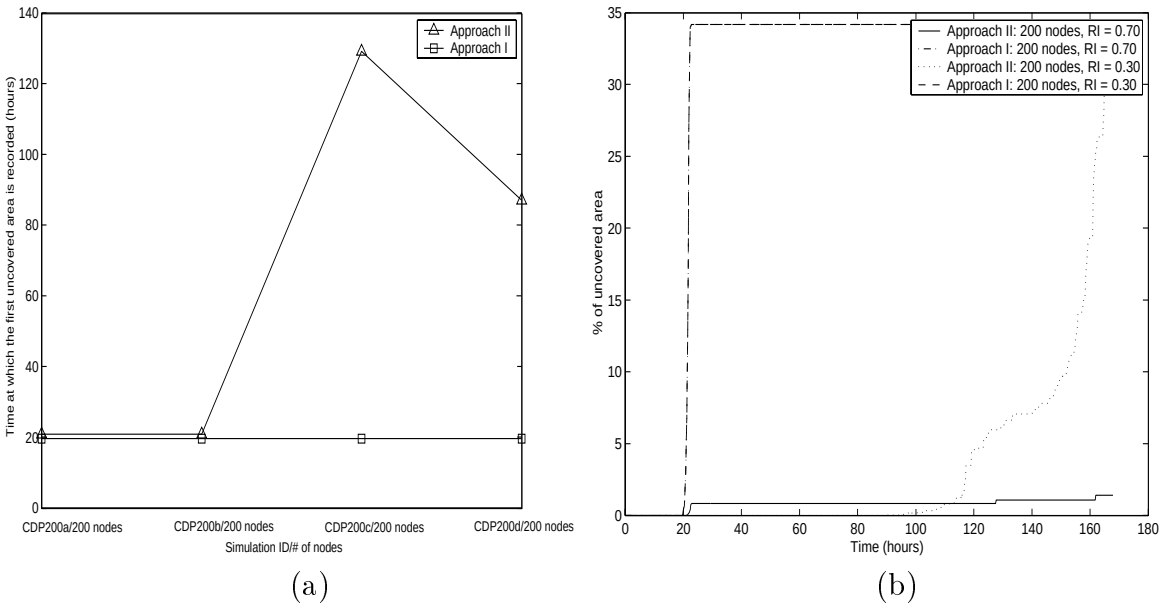


Figure 14: (a): Time at which the first uncovered area is recorded vs. Simulation ID. (b): Plots of % uncovered area vs. Time for different number of nodes.

the network. These plots indicate that while lower values of RI cause the node with the minimum energy to be protected, the overall coverage characteristic may not necessarily improve over time. Note that in all cases the Approach II exhibits performance superior to that of the Approach I.

6 Conclusions and Future work

This paper describes a set of distributed algorithms tasking distributed sensor networks in such a way as to conserve energy and balance the energy consumption load across the nodes. These algorithms were proven to be scalable with network size. Extensive simulation demonstrated the effectiveness of these algorithms in a variety of scenarios.

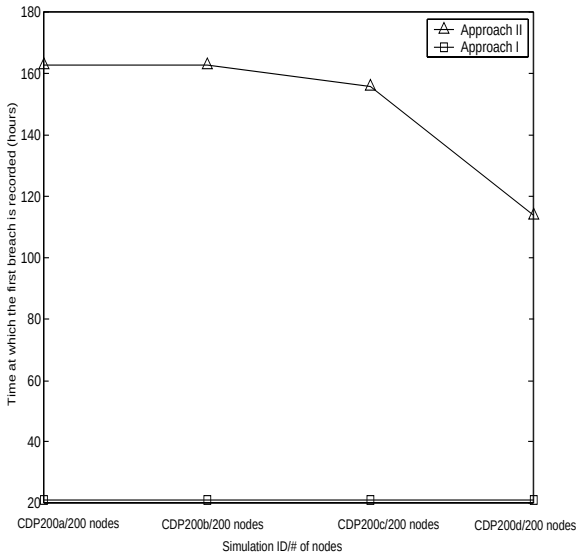


Figure 15: Time at which the first breach is recorded vs. Simulation ID.

The algorithms and simulations in this paper did not involve significant interaction with the underlying network routing scheme. This choice was made to achieve independence from any particular routing scheme. The underlying routing scheme will have an effect on the tasking algorithms; incorporating this interaction into the tasking algorithms will result in more effective tasking. For example, certain nodes may be critical to the function of the network; if these nodes are tasked like any other node, communications within the network may die prematurely. Further, the placement of sensor nodes was only touched on the experimental results. Sensor node placement can have a significant effect on the effectiveness of the tasking algorithms. Placement algorithms that take tasking into account could achieve further energy savings.

References

[1] M. Bhardwaj, T. Garnett, and A. P. Chandrakasan, *Upper bounds on the lifetime of sensor networks*, Proceedings. ICC 2001, (June 2001).

[2] K. Bult, A. Burstein, D. Chang, M. Dong, and W. Kaiser, *Wireless integrated microsensors*, Proceedings of Conference on Sensors and Systems (Sensors Expo). Anaheim, CA, USA, (April 16-18 1996), pp. 33–38.

[3] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, *Next century challenges: Scalable coordination in sensor networks*, ACM MobiCom 99, (1999).

[4] R. Gjertsen, M. T. Jones, and P. E. Plassmann, *Parallel heuristics for improved, balanced graph colorings*, Journal of Parallel and Distributed Computing, 37 (1996), pp. 171–186.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin, *Directed diffusion: A scalable and robust communication paradigm for sensor networks*, Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000), (2000).

[6] D. Jayasimha, D. Nadig, and S. Iyengar, *A versatile architecture for the distributed sensor integration problem*, Computers, IEEE Transactions on, 43 (February 1994), pp. 175–185.

[7] M. T. Jones and P. E. Plassmann, *A parallel graph coloring heuristic*, SIAM Journal on Scientific Computing, 14 (1993), pp. 654–669.

[8] J. Kahn, R. Katz, and K. Pister, *Next century challenges: mobile networking for smart dust*, In Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking, (1999), pp. 271–278.

[9] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM Journal of Computing, 15 (1986), pp. 1036–1053.

[10] H. O. Marcy, J. R. Agre, C. Chien, L. P. Clare, N. Romanov, and A. Twarowski, *Wireless sensor networks for area monitoring and integrated vehicle health management applications*, Collection of Technical Papers, AIAA Guidance, Navigation, and Control Conference and Exhibit,Portland, OR, 1 (Aug. 9-11, 1999).

[11] R. Min, M. Bhardwaj, S.-H. Cho, A. Sinha, E. Shih, A. Wang, and A. Chandrakasan, *Low-power wireless sensor networks*, VLSI Design 2001, (January 2001).

[12] R. Min, M. Bhardwaj, S.-H. Cho, A. Sinha, E. Shih, A. Wang, and A. P. Chandrakasan, *An architecture for a power-aware distributed microsensor node*, IEEE Workshop on Signal Processing Systems (SiPS '00), (October 2000).

[13] Opnet, *Opnet v6.0 Reference Manual*.

[14] K. Pister, J. Kahn, and B. Boser, *Smart dust: Wireless networks of millimeter-scale sensor nodes*, Highlight Article in 1999 Electronics Research Laboratory Research Summary, (1999).

[15] G. Pottie, *Wireless sensor networks*, 1998 Information Theory Workshop, Killarney, Ireland, (22-26 June 1998), pp. 139–40.

[16] G. Pottie and W. Kaiser, *Wireless integrated network sensors*, Communications of the ACM, 43 (May 2000), pp. 551–8.

[17] J. Rabaey, J. Ammer, J. da Silva Jr., and D. Patel, *Picoradio:ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes*, WVLSI 2000. Proceedings. IEEE Computer Society Workshop on, (2000), pp. 9–12.

[18] A. Savvides, S. Park, and M. B. Srivastava, *On modeling networks of wireless microsensors*, Technical Report TM-UCLA-NESL-2000-11-001, University of California, Los Angeles, November 2000.

[19] B. Schott, *Personal communication*, 2000.

[20] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, *A self-organizing wireless sensor network*, 37th Allerton Conference on Communication, Control, and Computing, (1999).

[21] M. Stemm and R. Katz, *Measuring and reducing energy consumpation of network interfaces in hand-held devices*, Transactions on Communications, Special Issue on Mobile Computing, 8 (1997), pp. 1125–1131.

[22] Y. Xu, J. Heidemann, and D. Estrin, *Adaptive energy-conserving routing for multihop ad hoc networks*, Research Report 527, USC/Information Sciences Institute, October 2000.

Appendix D:

SIMULATING NETWORKS OF WIRELESS SENSORS

Sung Park
Andreas Savvides
Mani B. Srivastava

Networked Emebedded Systems Laboratory
Electrical Engineering Departments
University of California, Los Angeles
Los Angeles, CA 90095, U.S.A.

ABSTRACT

Recent advances in low-power embedded processors, radios, and micro-mechanical systems (MEMs) have made possible the development of networks of wirelessly interconnected sensors. With their focus on applications requiring tight coupling with the physical world, as opposed to the personal communication focus of conventional wireless networks, these wireless sensor networks pose significantly different design, implementation, and deployment challenges. In this paper, we present a set of models and techniques that are embodied in a simulation tool for modeling wireless sensor networks. Our work builds up on the infrastructure provided by the widely used ns-2 simulator, and adds a suite of new features and techniques that are specific to wireless sensor networks. These features introduce the notion of a sensing channel through which sensors detect targets, and provide detailed models for evaluating energy consumption and battery lifetime.

1 INTRODUCTION

The marriage of ever tinier and cheaper embedded processors and wireless interfaces with micro-sensors based on micro-mechanical systems (MEMS) technology has led to the emergence of ***wireless sensor networks*** as a novel class of networked embedded systems. Many interesting and diverse applications for these systems are currently being explored. In indoor settings, sensor networks are already being used for condition-based maintenance of complex equipment in factories. In outdoor environments, these networks can monitor natural habitats, remote ecosystems, endangered species, forest fires, and disaster sites.

The primary interest in wireless sensor networks is due to their ability to monitor the physical environment through ad-hoc deployment of numerous tiny, intelligent, wirelessly networked sensor nodes. Because of the large numbers of sensor nodes required, and the type of applications sensor networks are expected to support, sensor nodes should be small, tetherless, and low cost. Due to these requirements, networked sensors are very constrained in terms of energy, computation and communication. The small form factor requirement prohibits the use of large long lasting batteries. Low production costs and low energy requirements suggest the use of small, low power processors, and small radios with limited bandwidth and transmission ranges. The ad-hoc deployment of sensor nodes implies that the nodes are expected to perform sensing and communication with no continual maintenance and battery replenishment. The energy constraints call for power awareness, which in turn leads to additional tradeoffs. The high-energy costs associated with wireless transmission, made particularly severe for sensor networks because nodes with small antenna heights placed on the ground see $1/r^4$ wireless link path loss coupled with the ever reducing cost of processing has led to a the adoption of a distributed computing viewpoint for wireless sensor networks. Instead of simply sending the raw data (perhaps compressed) to a gateway node, in typical applications the nodes in wireless sensor networks perform computation for decision making within the network, either individually via techniques such as signature analysis or in local clusters using coherent combining of raw sensor signals (i.e. beam forming) or non-coherent combining of decisions (i.e. Bayesian data fusion). By performing the computation inside the network, communication may be reduced thus prolonging the network lifetime

We construct a versatile environment in which sensor networks can be studied. This environment employs a wide range of models to orchestrate and simulate realistic scenarios. Furthermore, since power consumption is also a key design factor, we emphasize power consumption and battery behavior models. First we create a set of sensor node models that are derived from the empirical power characterization of two different nodes representing two extremes; the WINS node (Rockwell Scientific Company LLC. 2001) from Rockwell Science Center and the Medusa node, an experi-

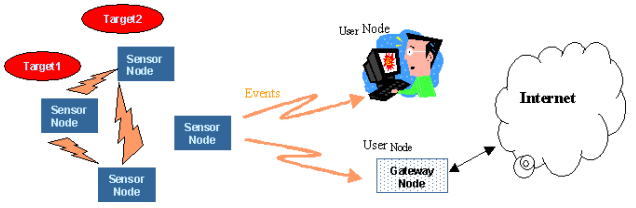


Figure 1: Sensor Network Scenario

mental prototype that we have constructed. These sensor node models are combined into the widely used event queue based network simulator, ns-2 (ns-2 Simulator 2001). By introducing the notion of *sensing channels* in our simulation environment and a flexible and highly parametrizable scenario generation tool, we can study the power consumption of sensor nodes by instrumenting complex sensor network scenarios in a detailed graphical environment.

2 RELATED WORK

Although sensor networks have recently received a lot of attention, there are still not many formal tools available for the systematic study of sensor networks. The work in (Ulmer 2001) presents a Java based simulator for sensor networks. This is an online simulator that can create and simulate simple topologies but does not have any explicit models for sensors or power management. Up to this point there is no publication on this work. On the network simulation, numerous simulators are currently available such as GloMoSim, OPNET and ns-2. These simulators provide great flexibility in the simulation of wireless ad-hoc networks at all layers. Despite their effectiveness, these tools are currently not equipped for capturing all the aspects of interest in sensor networks.

3 SIMULATION ARCHITECTURE OVERVIEW

We motivate our discussion with an example of a sensor network illustrated in figure 1. In this example, a set of wireless nodes equipped different sensor for monitoring natural habitat. The results of these sensors are processed within the network and the final sensing report is forwarded via wireless links to the gateway nodes that makes the results available on the internet. The main goal of our work is to recreate such scenarios in a versatile simulation environment where the behavior of the sensor network can be analyzed.

In our simulation environment, a typical sensor network scenario will consist of three types of nodes: 1) ***sensor nodes*** that monitor their immediate environment, 2) ***target nodes*** that generate the various sensor stimuli that are received by multiple sensor nodes via potentially many different transducers (e.g. seismic, acoustic, infrared) over different sensor channels; e.g. a moving vehicle generates ground vibrations that trigger seismic sensors and sound that triggers acoustic sensors, and 3*)* ***user nodes*** that represent clients and administrators of the sensor network. Shown in figure 2, three type of node models make up the key building blocks of our simulation environment. The sensor nodes are the key active elements, and form our focus in this section. In our model, each sensor node is equipped with one wireless network protocol stack and one or more sensor stacks corresponding to different types of transducers that a single sensor node may possess. The role of the sensor protocol stacks is to detect and process sensor stimuli on the sensing channel and forward them to the application layer which will process them and eventually transmit them to a user node in the form of sensor reports. In addition to the protocol and sensor stacks that constitute the algorithmic components, each node is also equipped with a power model corresponding to the underlying energy-producing and energy-consuming hardware components. This model is composed of an energy provider (the battery) and a set of energy consumers (CPU, Radio, Sensors). The energy consuming hardware components can each be in one of several different states or modes, with each mode corresponding to a different point in performance and power space. For example, the radio may be in sleep mode, receive mode, or one of several different

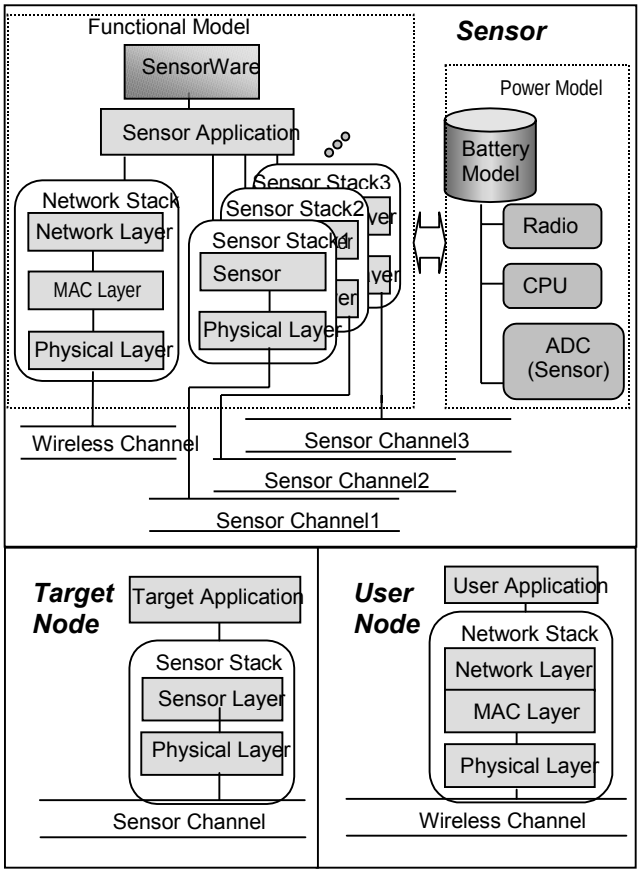


Figure 2: Sensor Node Model Architecture

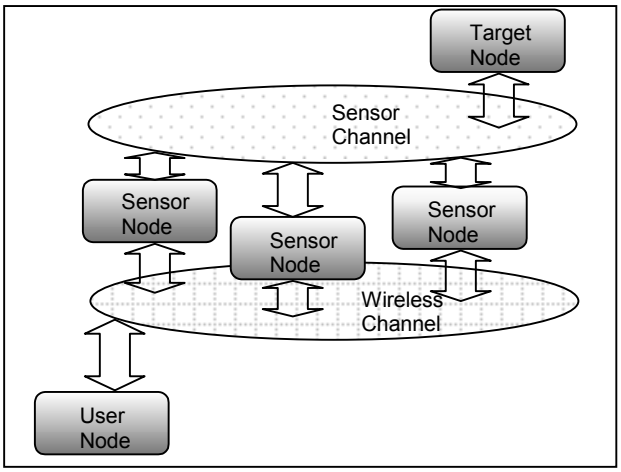


Figure 3: Sensor Network Model Architecture

transmit modes corresponding to different symbol rates, modulation schemes, and transmit power. Similarly, the CPU may be in sleep mode, or one of several different active modes corresponding to different frequency and voltage. The algorithms in the network and sensor stack control the change in mode of the power consumers. For example, the MAC protocol may change the radio mode from sleep to receive. In return, the performance of the algorithms may depend on the mode. For example, the time taken by the physical layer in the network protocol stack would depend on the data rate of the mode the radio is currently in. All of this is accomplished by having the algorithms in the network and sensor stacks issue mode change events to the power consumer entities, and having the algorithms read relevant parameter values from those entities. Algorithm-induced changes in the operating modes of power consuming hardware entities in turn affect the current drawn by them from the battery which delivers the power corresponding to the sum of current (or power) drawn by each power consumer. Internally, the battery entity depletes its stored chemical energy according to the efficiency dictated by the battery model.

Figure 3 illustrates how a typical sensor network will be constructed and simulated using our simulation environment. In figure 3, the wireless channel and sensor channel form separate communication mechanisms where events from different nodes are passed through. A typical scenario will involve a target node passing through a group of sensor nodes deployed in the field. As the target node moves around, it gives out sensor signal in the form of events through the sensor channel and each sensor node detects the events based on propagation model implemented in each node's sensor stack. When sensor nodes determine the sensor signals (events) are noteworthy, they transmit packets (also in the form of events) through the wireless channel destined to the user node.

By separating the sensor channel and the wireless channel, our sensor network model makes easier to simulate and analyze the operation of sensor network where the sensor signal detection events and wireless communication events can be received or transmitted concurrently. Moreover, by allowing sensor node to connect to multiple sensor channels, our simulation environment provides ability to analyze complex behaviors of sensor nodes' reaction to multiple sensor signals (i.e. seismic vibration, sounds, temperature, etc..) that can be detected all at the same time. In the following section, we discuss each components of the sensor node model shown in figure 2, and explain how we construct the model of different sensor nodes' components.

4 FRAMEWORK OF SENSOR NETWORK SIMULATION

4.1 Node Placement and Traffic Generation

In studying the performance of a wireless sensor network for a given application, a crucial element is the overall deployment scenario which includes the node placement topology, the radio ranges, the sensing ranges, the trajectories of the targets and resultant event traffics, and the trajectories of the user nodes and their query traffics. All these elements contribute to the different design trade-offs that can be made, and it is crucial to evaluate the effects of a new algorithm or protocol under diverse deployment scenarios.

To study such effects, we have developed a detailed scenario generation and visualization tool that enables us to construct detailed topologies and sensor network traffic. Our simulation environment enables us to assess the requirements of a sensor network under different circumstances by generating detailed scenario input to our simulations. This complements the scenario generation techniques provided in (ns-2 Simulator 2001) which are mainly targeted to ad-hoc wireless communication networks. Sensor node placement can vary depending on intended the task on the network. For example, to monitor wildlife in a forest, sensors may be uniformly distributed in the forest. If however, the sensor network is deployed for perimeter defense, then the sensors will most likely be distributed around a specified perimeter in a two dimensional gaussian distribution. In some other cases, the sensors may be manually placed according to the requirements of the user.

Besides placement, the traffic requirements may be even more diverse. Sensor network traffic can be classified into 3 main types: 1) *user-to-sensor* traffic, which is a result of user commands and queries to the network, 2) *sensor-to-user* traffic, which consists of the sensor reports to the user and 3) *sensor-to-sensor* traffic, which includes collaborative signal-processing of sensor events in the network before they are reported to the user. The last type of traffic is the most complex, and it depends on the sensing method.

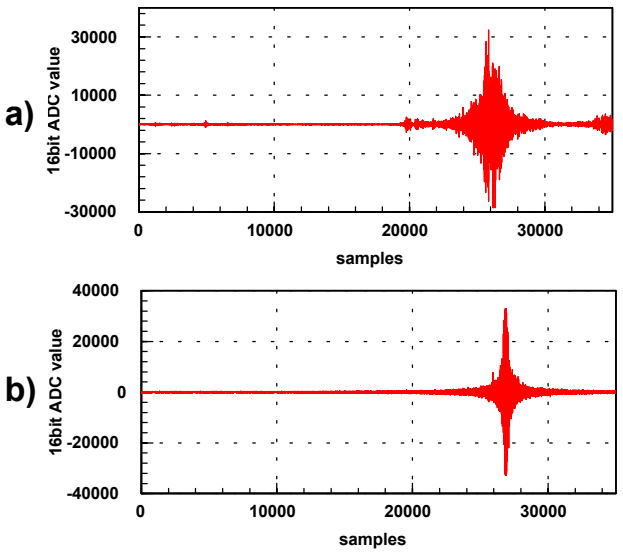


Figure 4: a) Real Target Seismic Signature b) Simulated Target Seismic Signature

4.2 Sensor Stack and Sensor Channel

The *sensor stack* simulates how a sensor node generates, detects and processes sensor signals. In sensor node model (figure 2), the *sensor stack* is a signal sink that is responsible for triggering the application layer every time a sensing event occurs. Various trigger functions ranging from simple sensing schemes to elaborate signal processing functions can be implemented in the sensor stack. In target node model, the *sensor stack* acts as a signal source. The sensor stack of a target node will contain a signature that is unique to the type of target the target node is modeling. The signature is then transmitted through various mediums (ground, air, free space, water, etc..) as the target node moves around. figure 4a and 4b show a real and a simulated signature obtained from a seismic sensor triggered by ground vibration from a traveling vehicle.

In figure 4, the ground is the medium that transmits the vibrations to the seismic sensor. We refer to this medium as the sensor channel, a model of a medium which sensor events such as seismic vibration, sounds, or infrared signals are traveled through. The type of medium can differ based on the type of sensor being modeled (seismic, acoustic, infra red, ultrasonic). Moreover, depending on the medium being modeled, the propagation of signal can differ. For instance, a sound moving through the air will have different propagation as the same sound moving through the water. In order to incorporate all these different aspects of the sensor network in to our simulation, we implement a simple sensor stack and sensor channel model by modeling the target node as a gaussian source whose signal amplitude is modeled as a gaussian random variable with the mean equal to zero and the variance σ^2. As the target travels through the sensor network, the target exerts the vibration signals (signal events) into the sensor channel periodically. The sensor channel then delivers this events to each sensor node's sensor stack and each sensor node adjusts the signal strength of the target based on sensor channels propagation model. The figure 4b demonstrates the signal strength variation as a target passes by a sensor node on a straight line. As the target approaches the sensor node, the signal strength increases, and as the target moves away, the signal attenuates rapidly. In this simulation the sensor signal was attenuated at a rate of 1/r where r is the distance between the target and the sensor.

4.3 Hardware Components Characterization

$$I_{sensor} = \frac{V_{test}}{R_{test}}$$

$$V_{Sensor} = V_{supply} - V_{test}$$

$$Power = I_{sensor} \times V_{sensor}$$

$$Energy = Power \times time$$

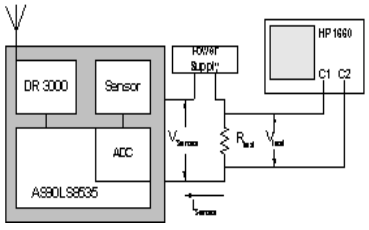


Figure 5: Power Measurement Configuration

Table 1: Experimental Node Current Consumption

Mode ID	CPU	Radio(OOK Modulation)	ADC	Total Current
1	Active 2.9mA	Tx-19.2kbps 5.2mA	On	8.1mA
2	Active 2.9mA	Tx-2.4kpbs 3.1mA	On	6.0mA
3	Active 2.9mA	Rx:4.1mA	On	7.0mA
4	Sleep 1.9mA	Sleep:5µA	On	1.9mA
5	Off 1µA	Sleep:5µA	Off	6µA

We construct our power models by performing measurements of the hardware power consumption using an HP 1660 oscilloscope, a bench power supply, and a high precision resistor. The measurement setup and power relationships are shown in figure 5. By characterizing each component of the sensor nodes we enable the simulated nodes to operate at different modes in which the power management schemes can switch different components on and off. Using the configuration in figure 5, the total current consumption of our experimental sensor node is obtained in Table 1. The measurements listed in Table 1 provide a better insight into the power consumption of the sensor nodes since the actual power consumption is oftentimes different from the typical values provided in the manufacturer data sheets depending on the mode of operation.

4.4 Battery Models

The Battery Model simulates the capacity and the lifetime of the sole energy source of the sensor node, the battery. In reality, battery behavior highly depends on the constituent materials and modeling this behavior is a difficult task. Although the battery can be viewed as a energy storage, the main goal of the sensor network is to increase the lifetime of the battery. Thus, in this section, we focus on how battery's capacity can be modeled based on the energy consumers' behavior. We propose 3 different types of battery models to study how different aspects of real battery behavior can affect the energy efficiency of different applications. The metrics that are used to indicate the maximum capacity of the battery is in the unit of Ah (Ampere*Hour). The metric is a common method used by the battery manufacturers to specify the theoretic total capacity of the battery. Knowing the current discharge of the battery and the total capacity in Ah, one can compute the theoretical lifetime of the battery using the equation , $T = \frac{C}{I}$, where T=battery lifetime, C=rated maximum battery capacity in Ah, and I=discharge current.

4.4.1 Linear Model

In Linear Model, the battery is treated as linear storage of current. The maximum capacity of the battery is achieved regardless of what the discharge rate is. The simple battery model allows user to see the efficiency of the user's application by providing how much capacity is consumed by the user. The remaining capacity after operation duration of time t_d can be expressed by the following equation.

$$\text{Remaining capacity (in Ah)} = C = C' - \int_{t=t_0}^{t_0+t_d} I(t)dt \qquad (1)$$

where C' is the previous capacity and $I(t)$ is the instantaneous current consumed by the circuit at time t. Linear Model assumes that $I(t)$ will stay the same for the duration t_d, if the operation mode of the circuit does not change (i.e. radio switching from receiving to transmit, CPU switching from active to idle, etc..) for the duration t_d. With these assumptions equation 1 simply becomes as the following.

$$C = C' - \int_{t=t_0}^{t_0+t_d} I(t)dt = C' - I \cdot t\Big|_{t_0}^{t_0+t_d} = C' - I \cdot t_d \qquad (2)$$

The total remaining capacity is computed whenever the discharge rate of the circuit changes.

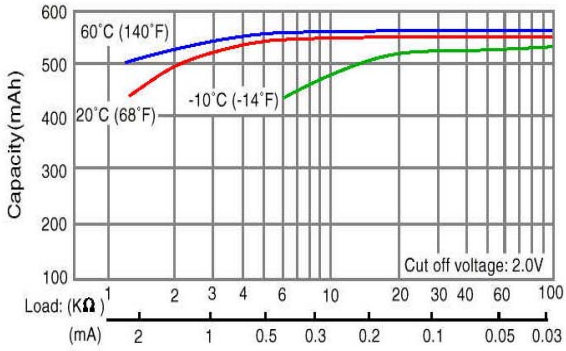


Figure 6: Capacity vs. Discharge Rate Curve for CR2354 (Matsushita Electric Corp. of America 2001)

4.4.2 Discharge Rate Dependent Model

While Linear Model assumes that the maximum capacity of the battery is unaffected by the discharge rate, Discharge Rate Dependent Model considers the effect of battery discharge rate on the maximum battery capacity. In [15] [16], it is shown that battery's capacity is reduced as the discharge rate increases. In order to consider the effect of discharge rate dependency, we introduce factor k which is the battery capacity efficiency factor that is determined by the discharge rate. The definition of k is, $k = \frac{C_{eff}}{C_{\max}}$, where C_{eff} is the effective battery capacity and C_{max} is the maximum capacity of the battery with both terms expressed in unit of Ah. In Discharge Rate Dependent Model, the equation 1 is then transformed to the following.

$$C = k \cdot C' - I \cdot t_d \qquad (3)$$

The efficiency factor k varies with the current I and is close to one when discharge rate is low, but approaches 0 when the discharge rate becomes high. One way to find out corresponding k value is for different current value of I is to use the table driven method introduced in (Simunic 1999).

4.4.3 Relaxation Model

Real-life batteries exhibit a general phenomenon called "relaxation" explained in (Fuller 1994, Linden 1995, Chiasserini 1999). When the battery is discharged at high rate, the diffusion rate of the active ingredients through the electrolyte and electrode falls behind. If the high discharge rate is sustained, the battery reaches its end of life even though there are active materials still available. However, if the discharge current from the battery is cutoff or reduced during the discharge, the diffusion and transport rate

of active materials catches up with the depletion of the materials. This phenomenon is called relaxation effect, and it gives the battery chance to recover the capacity lost at high discharge rate. For a realistic battery simulation, it's important to look at the effects of relaxation as it has effect of lengthening the lifetime of the battery. For our simulation, we adapt the analytical model introduced in (Fuller 1994) which takes discharge rate as input and computes the battery voltage over the simulation duration.

5 EXAMPLE STUDY CASE

In this section we demonstrate some of the main capabilities of our tool by studying the performance of different battery models with various sensor node operation profile.

5.1 Low Rate/Low Power vs. High Rate/High Power

In this case study, we evaluate the battery consumption of our experimental sensor node by considering different operation profiles. In section 4.3, we have discussed how each component of our sensor node has different power consumption depending on its operation mode. In this section, we examine how the combination of the operation modes of different components affects the aggregate power consumption of the sensor node. The scenario involves two sensor nodes (a transmitter and a receiver) that are within the transmission range of each other (approximately 15 meters apart) where the transmitter needs to transmit a 2MB file to the receiver. For the purposes of our discussion we define 5 different operation modes for our experimental node shown in table 1. To examine the energy consumption and communication tradeoffs we evaluate 3 different data transmission policies.

1)19.2 kbps continuous transmission: The transmitter sends data at the highest data rate without any break. The transmitter will be operating in mode 1 and the receiver will be operating in mode 3; 2) 2.4 kbps continuous transmission: With lower data rate the sender can transmit at a lower power level to reach the receiver. The transmitter will be operating in mode 2 and the receiver will be operating in mode 3; 3) 19.2 kbps pulse transmission: The transmitter sends data intermittently at the highest power level. While the transmitter is not transmitting, the transmitter puts the CPU and Radio to sleep. The transmitter power cycle its component by transmitting one 60 byte packet at 19.2 kbps for .025 sec and sleeps for .125 sec until all the data is received by the receiver. The transmitter will be switching between modes 1 and 4, and the receiver will be switching between mode 3 and 4.

Figure 7a shows the effect on each battery model capacity after the 2 MB data transfer for the three transmission methods described above. This experiment was performed for all three battery models described in section 4.4. Initially, all batteries were set to a capacity level of 10 mA*hour. The left half of figure 7a describes the remaining battery capacity of the transmitter after the file transfer, and the right half shows the receiver battery capacity. The solid bar in the figure indicates the total time for data transfer. Looking at the solid bar, it is clear that the sending the file at high data rate takes the least time thus the least battery capacity. Although the 2.4 kbps transmission and 19.2 kbps transmission took the same amount of time to transmit the data, 19.2 kbps pulse transmission saved much battery capacity due to the sleep period. Figure 7a also shows how different battery models exhibit different characteristics under different transmission methods. The linear model shows how optimum battery will behave as it shows the theoretical capacity of the battery under any discharge current. On the other hand, the rate dependent model accurately describes how real batteries will behave when there is a constant discharge for long duration. This is shown in 2.4 kbps transmission where the remaining capacity of rate dependent model is substantially less than the

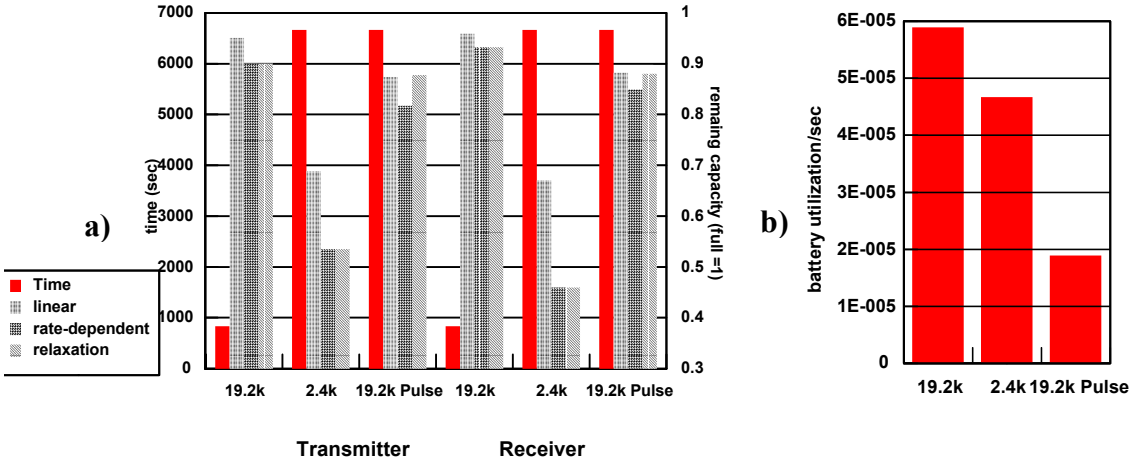


Figure 7: a) Battery Capacity Usage Under Different Discharge Profiles, b) Battery Utilization Rate

linear model. The other interesting model is the relaxation model which exhibits the both discharge rate dependent capacity and recovery effect. Since the relaxation model has recovery properties, the difference between relaxation model and rate dependent model is shown in the pulsed transmission cases. In figure 7a the relaxation model has the same remaining battery life as rate dependent model for 19.2kbps and 2.4 kbps continuous transmission and reception. However, in 19.2 kbps pulse transmission and reception, the relaxation model has almost equal capacity as the linear model due to the capacity recovery during the sleep mode.

5.2 Monitoring a Moving Vehicle in a Sensor Field

In this implementation we first show the effect of traffic on the sensing and communication traffic and then we evaluate simple power management scheme using the same sensor node setup as in the previous subsection. For this we have implemented a lightweight protocol stack similar to what one would expect to have on a tiny sensor node. The radio transmission and reception are driven by a TDMA based medium access control (MAC) protocol based on unique slot assignment algorithm derived from [9]. The MAC protocol assigns a unique slot to each node over a 2-hop radius and each node is aware of its one-hop neighbors and their corresponding slot assignment. For routing, we have implemented a very lightweight table-driven routing protocol with table size of one (next hop to user node). The motivation for TDMA scheme comes from our result in section 5.1 where a pulse transmission and reception can improve the battery utilization in the long term. In our power aware TDMA scheme this requirement is met for both the transmission and reception of packets. For transmission, a node is only allowed to transmit in its assigned time-slot. For reception, a node only needs to listen to the wireless channel for the duration of the slots that are already assigned to its one-hop neighbors. For the purposes of our discussion we refer to all the other remaining slots as *idle slots*.

In this scenario, a small cluster of 10 sensors equipped with seismic sensors is deployed to detect a bypassing truck as shown in figure 8a. The seismic sensors run at a sample rate of 400Hz to produce 16 bit samples. The sensor nodes are configured to report back to a gateway node that makes the results available on the Internet. Each sensor is programmed to transmit a report to the gateway within 5 seconds from the moment the ground vibrations from the truck are detected. If at least 2048 samples are obtained, the node can perform coherent detection and it will transmit a 10 byte to report the target type. This short packet is called "coherent traffic". If however, the node does not have enough samples at the end of the 5-second period, it enters a non-coherent detection mode and transmits all its available samples to the gateway node which performs sensor data combination (called "beamforming" [20]) to improve the detection accuracy. Since the sensor node transmits raw data when it enters non-coherent detection mode, the size of non-coherent data tends to be lot larger than the coherent traffic. This non coherent raw data is referred to as "non-coherent traffic" During the simulation, the network traffic will be consist of coherent and non-coherent traffic depending on whether the individual sensors successfully classified the target. We discuss the result of the simulation in the following two sub sections.

5.2.1 Efficiency of Power Management Scheme with TDMA

One apparent advantage of TDMA over other CSMA random access MAC protocols is the fact that the sensor nodes do not have to be in receive mode during the time slots where none of its neighbors are schedule to transmit. This allows the sensor nodes to perform a simple power management scheme that puts the CPU and radio to sleep during *idle slots* to conserve battery capacity. With this setup, we evaluate the efficiency of battery capacity utilization when this simple power management scheme is used. Figure 8a is the scenario used for our evaluation. It consists of 100 nodes uniformly distributed across a sensor field. The

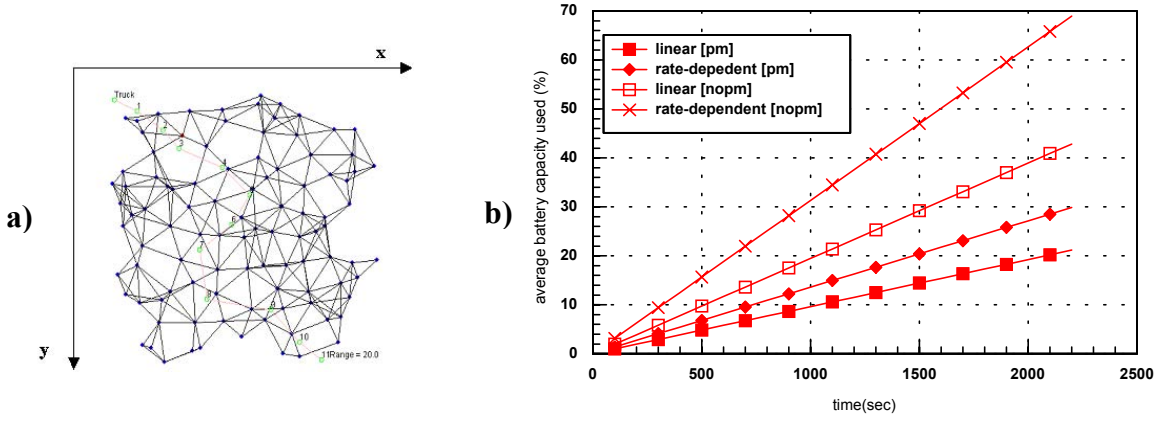


Figure 8: a) 100 Node Test Topology, b) Battery Capacity Usage

target travels at approximately 22 mph (10 m/s) through the track every 2 minutes. The target signals have an effective range of 20 meters. As the target travels through the sensor field, every node within the range of the target start collecting signal samples at 400 Hz, then send reports to the user node. We tested this scenario using the linear model and the rate dependent model by looking at battery utilization when the power management scheme is implemented (PM) and when there is no power management (NOPM).

The current drawn by each node will be similar to the cases described in section 5.1 with power management case resembling the 19.2 kbps pulse transmission and the no power management case resembling the 19.2 kbps continuous transmission. Figure 8b shows the average battery capacity utilization for each node. The bottom two curves show the difference in battery capacity utilization when the power management was used and the top two describe the cases when no power management is used. As the figure indicates, there is almost 100% improvement of battery utilization with the power management.

5.2.2 Effect of Sensor Power Cycle

In addition to the battery saving achieved by the TDMA power management scheme, we further look at how the sensor nodes can power cycle their sensors to conserve battery capacity. In this scenario (figure 9a), a square grid of sensor network is strategically placed over a flat field. The target travels along a pre-specified path and the sensor nodes attempts to make either coherent or non-coherent detection as described in the previous section. One difference in this scenario is that the sensor nodes attempt to turn on the sensors only intermittently to conserve power. When the sensor is turned off, the CPU of our experimental sensor node can go to mode 5 (table 1) where the power consumption is in the range of microwatts. However, the trade-off comes from the reduction of detection and classification accuracy since the sensor will miss the sensor signals coming from the target when they are turned off. The cost of such missed events may be very application specific. If the target occurrence is very frequent, it may be okay to miss its detection, but if the occurrence is very infrequent, it may be very crucial to detect that one incidence. It is possible that the whole sensor network may have been deployed to detect that "one" incidence. Therefore, in designing sensor network it's crucial to look at the application requirement as well as the target characteristics to guarantee of certain quality of service (QoS) similar to the one provided in telecommunication network. One such QoS guarantee will be something like "a target with a 20 mph speed following this track will not pass through the sensor field undetected". In this section, we try to look at what would be the maximum battery power saving that can be achieved while providing such QoS guarantees.

We look at the impact of a simple power management scheme which randomly wakes up the sensor within a pre-specified time window of 100 seconds and stay up for different percentage of duration. Figure 9b shows the battery capacity used and the amount of coherent data bytes transmitted for different power cycle durations. The plot indicates that there is a rapid decrease in coherent detection as the power cycle percentage decrease from 60% to 50%. On the other hand, the battery utilization steadily decreases as the power cycle percentage decreases.

6 CONCLUSIONS

We have demonstrated a flexible toolset for studying power consumption in sensor networks. With the flexible architecture that closely simulate the behavior of real sensor network, accurate power models of sensor nodes and

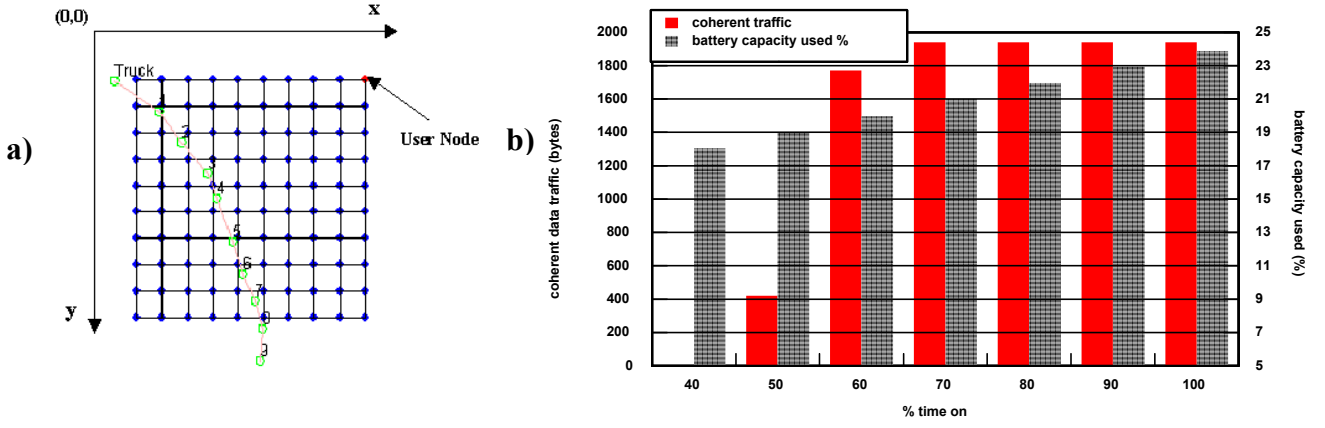


Figure 9: a) Sensor Scenario in Grid Sensor Network b) Coherent Traffic Data and the Battery Efficiency Of Various Power Cycle Durations

analysis of battery behavior are utilized in a tool to evaluate power consumption in the context of a realistic scenario. With these results we can assess the power consumption for new sensor nodes that are currently under development. Furthermore, this tool has been an indispensable aid in estimating the resources required for the network protocols to function correctly in new node architectures. By simulating and validating target protocols we can also get a good indication of code size and memory requirements thus resulting in feasible low cost designs. We envision that this set of tools will play an instrumental role in the design and implementation of new application specific sensor networks.

REFERENCE

Chiasserini, C. F., and R. R. Rao. 1999. Pulsed battery discharge in communication devices. In *Proceedings of Mobicom 99*, Seattle, August 1999.

Fuller, T. F., M. Doyle, and J. Newman. 1994. Simulation and Optimization of the Dual Lithium Ion Insertion Cell. *Journal of Electrochem. Soc*., vol. 141, no. 4, pp 1-10.

Linden, H.D. 1995. *Handbook of Batteries*. 2nd ed. New York: McGrawHill.

Matsushita Electric Corp. of America. 2001. *Panasonic lithium coin cell battery datasheet*. Available via `http://www.panasonic.com/industrial_oem/battery/battery_oem/chem/lith/lith.htm` [accessed July 9, 2001].

ns-2 simulator. 2001. *ns-2 Simulator*, Available via `http://www.isi.edu/nsnam/ns/` [accessed July 9, 2001].

Rockwell Scientific Company LLC. 2001. *WINS (Wireless Integrated Network Systems) Project*. Available via `http://wins.rsc.rockwell.com/` [accessed July 9, 2001].

Simunic, T., L. Benini, and G. De Micheli. 1999. Energy-Efficient Design of Battery-Powered Embedded Systems. In *Proceedings of International Symposium on Low Power Electronics and Design*, 212-217, Piscataway, New Jersey.

Ulmer, C. 2001. *Sensor Network Simulator*, Available via `http://users.ece.gatech.edu/~grimace/research/sensorsimii/` [accessed July 9, 2001].

AUTHOR BIOGRAPHIES

SUNG PARK is a Ph.D. student of Electrical Engineering at University of California Los Angeles. He received his MS from Carnegie Mellon University in 1997. His research interests are network simulation and modeling, low power embedded systems, and sensor network. His email and web addresses are <spark@ee.ucla.edu> and <http://www.ee.ucla.edu/~spark>.

ANDREAS SAVVIDE is a Ph.D. student of Electrical Engineering at University of California Los Angeles. He received his MS from University of Massachusetts, Amherst in 1997. His research interests are adhoc wireless network, embedded system, and sensor network. His email and web addresses are <asavvide@ee.ucla.edu> and <http://www.ee.ucla.edu/~asavvides>.

MANI B. SRIVASTAVA is an Associate Professor of Electrical Engineering at University of California Los Angeles. He received his Ph.D. from U. C. Berkeley in 1992. His research interests are low power VLSI circuits, embedded systems, and sensor networks. His email and web addresses are <mbs@ee.ucla.edu> and < http://nesl.ee.ucla.edu/people/mbs/>.

Appendix E:

ENERGY EFFICIENT ROUTING IN WIRELESS SENSOR NETWORKS

Curt Schurgers
Mani B. Srivastava

Networked & Embedded Systems Lab (NESL), Electrical Engineering Department
University of California at Los Angeles (UCLA), CA

ABSTRACT

Wireless sensor nodes can be deployed on a battlefield and organize themselves in a large-scale ad-hoc network. Traditional routing protocols do not take into account that a node contains only a limited energy supply. Optimal routing tries to maximize the duration over which the sensing task can be performed, but requires future knowledge. As this is unrealistic, we derive a practical guideline based on the energy histogram and develop a spectrum of new techniques to enhance the routing in sensor networks. Our first approach aggregates packet streams in a robust way, resulting in energy reductions of a factor 2 to 3. Second, we argue that a more uniform resource utilization can be obtained by shaping the traffic flow. Several techniques, which rely only on localized metrics are proposed and evaluated. We show that they can increase the network lifetime up to an extra 90% beyond the gains of our first approach.

I. INTRODUCTION

Recently IC and MEMS have matured to the point where they enable the integration of communications, sensors and signal processing all together in one low-cost package. It is now feasible to fabricate ultra-small sensor nodes that can be scattered on the battlefield to gather strategic information [1]. The events detected by these nodes need to communicated to gateways or users who tap into the network. This communication occurs via multi-hop routes through other sensor nodes. Since the nodes need to be unobtrusive, they have a small form-factor and therefore can carry only a small battery. As a result, they have a limited energy supply and low-power operation is a must. Multi-hop routing protocols for these networks necessarily have to be designed with a focus on energy efficiency.

The proposed approaches lean towards localized algorithms [1][2]. Due to the large number of sensors, network-scale interaction is indeed too energy expensive. Moreover, a centralized algorithm would result in a single point of failure, which is unacceptable in the battlefield. In this paper, we propose two options for localized algorithms to increase the sensor network lifetime: (1) minimize the energy consumption of transmissions and (2) exploit the multi-hop aspect of network communications.

The first option is to combine/fuse data generated by different sensors [1][2]. In [3] cluster head selection is proposed to perform this task. However, in section IV, we present a robust way of achieving the same functionality without explicit cluster formation.

The second option focuses on the paths that are followed during the data routing phase. The framework presented in [2] advocates a localized model called ‘directed diffusion’. Other work uses information on battery reserve and the energy cost to find the optimal routes [4]. The routing protocol in [5] is based on the node’s location, transmit energy and the residual battery capacity. In contract to this prior work, we propose a guideline that aims at spreading the network traffic in a uniform fashion. Our spreading ideas, although partly tailored towards the underlying routing algorithm we have chosen, should be beneficial for the energy aware routing protocols mentioned above. We discuss these spreading techniques in section V.

However, before discussing our data fusion and spreading, we first focus on the problem statement: how to increase the lifetime of a network of energy constrained devices. This results in a practical guideline, which considers the energy histogram. All of this is treated in section II.

II. PROBLEM STATEMENT

1. Energy Optimal Routing

Traditional ad-hoc routing algorithms focus on avoiding congestion or maintaining connectivity when faced with mobility [6]. They do not consider the limited energy supply of the network devices. The example of figure 1 illustrates how the limited supply alters the routing issue. Nodes *A* and *E* first send 50 packets to *B*. Afterwards, *F* sends 100 packets to *B*. From a load balancing perspective, the preferred paths are *ADB*, *ECB* and *FDB* respectively.

However, when the nodes are energy constrained such that they can only send 100 packets, these paths are no longer optimal. Indeed, *D* would have used up 50% of its energy

before it can forward packets from *F* to *B*. In this case, all packets could have been delivered by choosing paths *ACB*, *ECB* and *FDB*. If, instead of *F*, node *C* would have become active, *A* should have used the original path *ADB*.

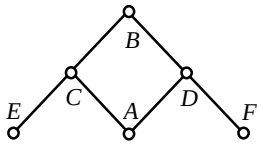


Figure 1: Load versus energy oriented routing

This simple case study highlights the following crucial observation: **optimal traffic scheduling** in energy constrained networks requires future knowledge. In our example, a maximum number of packets can reach *B* only if right from the start we know exactly when (and which) nodes will generate traffic in the future.

2. Energy Efficient Routing

Ideally, we would like the sensor network to perform its functionality as long as possible. Optimal routing in energy constrained networks is not practically feasible (because it requires future knowledge). However, we can soften our requirements towards a statistically optimal scheme, which maximizes the network functionality considered over all possible future activity. A scheme is **energy efficient** (in contrast to 'energy optimal') when it is statistically optimal and causal (i.e. takes only past and present into account).

In most practical surveillance or monitoring applications, we do not want any coverage gaps to develop. We therefore define the **lifetime** we want to maximize as the worst-case time until a node breaks down, instead of the average time over all scenarios. However, taking into account all possible future scenarios is too computationally intensive, even for simulations. It is therefore certainly unworkable as a guideline to base practical schemes on. Considering only one future scenario leads to skewed results, as shown in the example of figure 1.

3. Traffic Spreading Rationale

To derive a practical guideline, we start from the following observation: the minimum hop paths to a user for different streams tend to have a large number of hops in common [7]. Nodes on those paths die sooner and therefore limit the lifetime of the network. Figure 2 presents a typical energy consumption histogram at a certain point in time. Some nodes have hardly been used, while others have almost completely drained their energy.

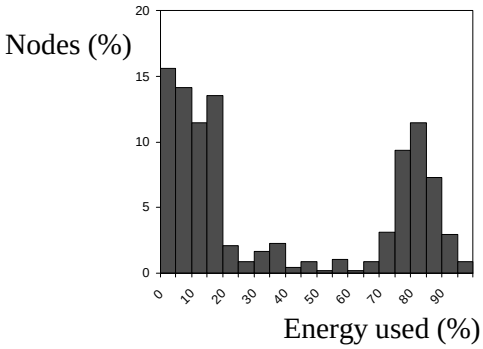


Figure 2: Undesirable energy histogam

As nodes that are running low on energy are more susceptible to die sooner, they have become more critical. If we assume that all the nodes are equally important (we revisit this assumption in section V.2), no node should be more critical than any other one. At each moment every node should therefore have used about the same amount of energy, which should also be minimized. The histogram of figure 3 is thus more desirable than the one of figure 2, although the total energy consumption is the same.

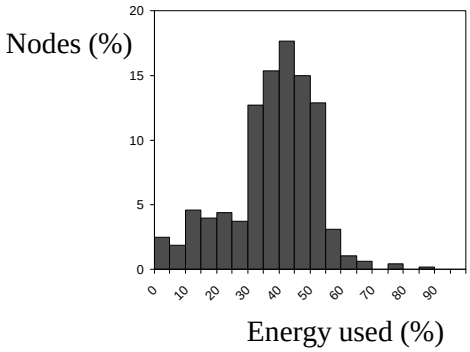


Figure 3: Desirable energy histogram

Striving for a compact energy histogram translates into the guideline that traffic should be spread over the network as uniformly as possible. Since visualizing the histogram over time is hard, we propose to use the root mean square E_{RMS} as an indicator instead (the lower this value, the better). It provides information on both the total energy consumption and on the spread.

III. BASIC ROUTING

As an underlying routing scheme, we base ourselves on the paradigm of directed diffusion [2]. When a user taps into the sensor network, he announces the type of information he is interested in. While flooding this 'interest' possibly using techniques like SPIN [8], gradients are established in each node. These gradients indicate the 'goodness' of the

different possible next hops and are used to forward sensor data to the user.

We have opted for a simple instantiation of this paradigm, which we call Gradient-Based Routing (GBR). While being flooded, the 'interest' message records the number of hops taken. This allows a node to discover the minimum number of hops to the user, called the node's **height**. The difference between a node's height and that of its neighbor is considered the gradient on that link. A packet is forwarded on the link with the largest gradient. Although our techniques to increase the network lifetime are built upon GBR, the main principles are general enough to also be applicable to other ad-hoc routing protocols.

IV. DATA COMBINING

1. Data Combining Entities (DCE)

Individual sensor nodes process their sensor data before relaying it to the user [1]. It is advantageous to combine observations from different nodes to increase the resource efficiency. This process reduces not only the header overhead, but also the data itself can be compacted as it contains partly the same information.

Although this combining can be implemented by explicitly selecting a cluster head [3], we present a scheme that is more robust to random node failures. First note that sensor nodes that are triggered by the same event, are typically located in the same vicinity. The resulting cloud of activated nodes is also in close communication proximity. The routes from these nodes to the user merge early on [7]. Nodes that have multiple streams flowing through them can create a Data Combining Entity (DCE), which takes care of the data compaction. Simulations have shown that the DCEs are located inside or very close to this cloud of activated nodes.

This scheme is highly robust. When a node with a DCE dies, the packets automatically take an alternative route and pass through another node that can create a new DCE.

2. Simulations

Figure 4 depicts the effects of our DCE-based data compaction on the total energy consumption. The nodes in this simulation are distributed randomly over a rectangular area with a constant width of 32 m and a linearly increasing length *B*. The radio transmission range *R* is 20 m and the average node density is kept constant at $10^{-2}/m^2$. The nodes at the top of this area sense a target and notify a user that is located at the bottom end (the transmission of one packet takes 5.76 ∝J). For our numerical results, we assume that a packet that is combined with another one can be compressed to 60% of its original size. We consider 3 distinct cases: without DCE, with at most one DCE (a compression bit in the packet header signals if the packet has been compressed already) on each route to the user and with no restrictions on the number of DCEs. The reduction in energy consumption is as expected (up to a factor 2 to 3), linearly proportional to the number of bits sent.

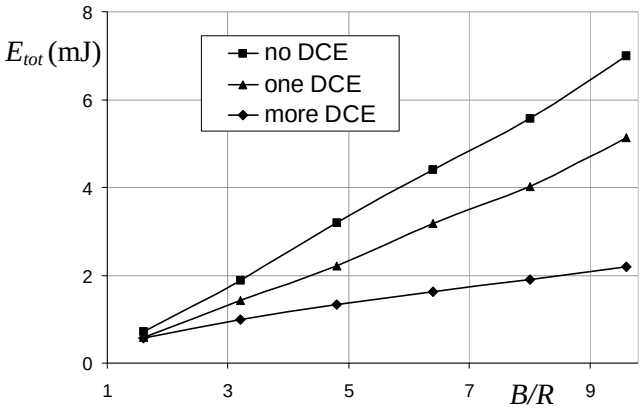


Figure 4: Energy comparison for DCE

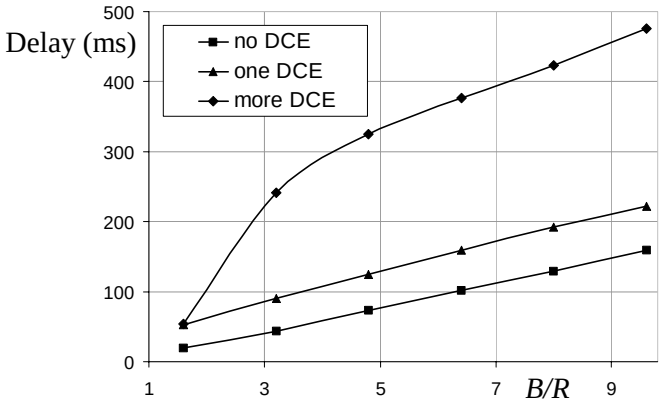


Figure 5: Delay comparison for DCE

The flip side is the average delay per packet, which is presented in figure 5. Since DCEs have to buffer data for a while, the packet delay will increase with the number of combining stages applied. Whether or not this is acceptable depends on the application.

V. NETWORK TRAFFIC SPREADING

1. Spreading Techniques

Stochastic Scheme: Using a rationale similar to the one of [9], each node can select the next hop in a stochastic fashion. More specifically, when there are two or more next hops with the same lowest gradient, a random one is

chosen. This does not increase the length of the path followed, but nonetheless contributes to spreading the network traffic.

Energy-based Scheme: When a node detects that its energy reserve has dropped below a certain threshold (50% in our simulations), it discourages others from sending data to it by increasing its height. This may change a neighbor's height (since a node's height is one more than that of its lowest neighbor). It in turn informs other nodes and these updates are propagated as far as is needed to keep all the gradients consistent.

Stream-based Scheme: The idea is to divert new streams away from nodes that are currently part of the path of other streams. A node that receives packets tells all its neighbors except to the one from where the stream originates, that its height has increased. Again, other nodes must make sure the gradients remain consistent. As a result of this scheme, the original stream is unaffected, since those nodes have not updated the height of the next hop. New streams of packets, however, will take other paths as the height of the nodes on the first path has apparently increased.

2. Simulations

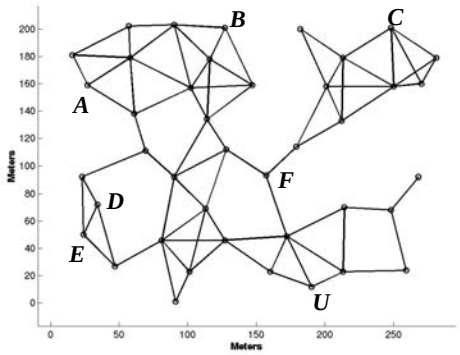


Figure 6: Wireless sensor network topology

Scenario 1: Nodes *A* and *B* (see figure 6) detect a different target and send packets to the user at regular intervals. After generating 100 packets each (this takes 11.8 seconds), these targets disappear and both nodes become inactive again. At this time, no node has been drained yet completely and the network connectivity is still fully intact. We have assumed a node has only 0.76 mJ of energy at its disposal (which is enough to send about 140 packets). The results can readily be scaled towards more realistic scenarios. Figure 7 shows the evolution of E_{RMS} as a function of time, for 5 different schemes. The unenhanced GBR is called 'standard'. Besides the three schemes discussed in V.1, we have also studied a combination of the stochastic and energy-based one.

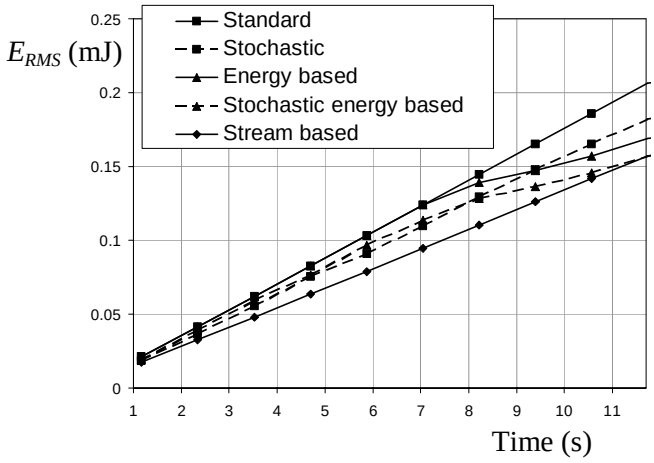


Figure 7: E_{RMS} for scenario 1

It is clear that the stream-based scheme indeed spreads the traffic more uniformly over the network. As soon as the energy of some nodes drops below 50%, the energy-based scheme kicks in. The stochastic routing provides an improvement both on top of the normal GBR and on top of the energy-based scheme.

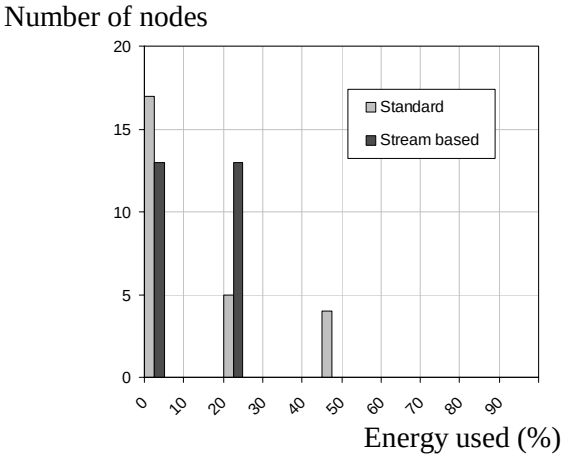


Figure 8: Energy histogram after 6 seconds

To verify that the E_{RMS} captures the relevant information, figure 8 shows the energy histogram for the standard and the stream-based scheme after 7 seconds. It is clear that spreading balances the energy consumption better.

Finally, we would like to show that the improved energy histogram is able to extend the network lifetime for a particular future scenario, although this does not prove anything about other possible futures. After 11.8 seconds node *C* starts forwarding packets to the user. Table 1 shows that the schemes that resulted in better traffic spreading also increase total traffic that reaches the user. We have verified that the time the network remains intact is increased by 90% when using the stream-based scheme.

Scheme	Packets received
Standard	127
Stochastic	133
Energy-based	160
Stochastic energy-based	161
Stream-based	175

Table 1: Packets received for scenario 1

Scenario 2: Nodes *D* and *E* (figure 6) each send 100 packets to the user in 11.8 s. Figure 9 illustrates that our traffic spreading schemes again result in a more uniform utilization of the network resources.

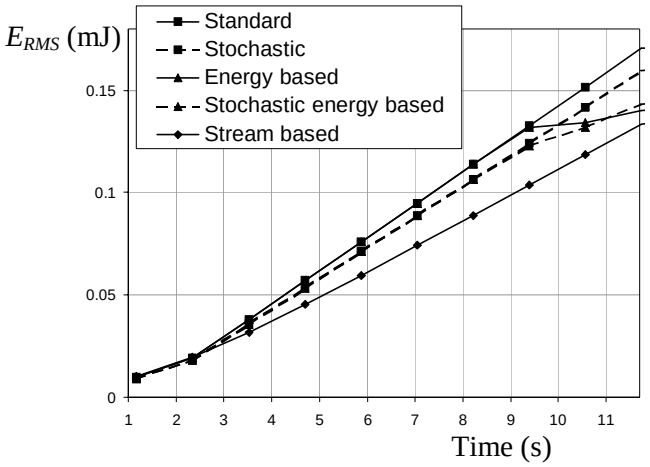


Figure 9: E_{RMS} for scenario 2

As before, we investigate one particular future activity scenario: node *C* becomes active after 11.8 seconds. From table 2, we conclude that spreading the network traffic has a negative effect as fewer packets are received! This is because the route taken by the standard GBR protocol avoids bottleneck node *F*. On the other hand, preading the traffic of *D* and *E* diverts some packets via *F* and therefore already partly drain this node before *C* can use it. This illustrates that spreading might increase the lifetime, although this does improve all possible futures. We observe however that the problems in this case are largely due to the fact that node *F* is critical as it is the only gateway to an entire subnet. Enhanced spreading techniques should therefore try to avoid critical nodes.

Scheme	Packets received
Standard	217
Stochastic	211
Energy-based	193
Stochastic energy-based	193
Stream-based	176

Table 2: Packets received for scenario 2

VI. CONCLUSIONS

In this paper we have argued that optimal routing in sensor networks is infeasible. We have proposed a practical guideline that advocates a uniform resource utilization, which can be visualized by the energy histogram. We acknowledge however that this is only a first cut at tackling this complicated issue. For example, exceptions must be made when nodes are critical in the overall network connectivity. We also propose a number of practical algorithms that are inspired by this concept. Our DCE combining scheme reduces the overall energy, while our spreading approaches aim at distributing the traffic in a more balanced way. We note that although we have started from GBR, our basic ideas and techniques should be able to enhance other routing protocols as well.

REFERENCES

[1] Sohrabi, K., Gao, J., Ailawadhi, V., Pottie, G., "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Communications Mag.*, Vol.7, No.5, pp.16-27, Oct. 2000.

[2] Estrin, D., Govindan, R., "Next Century Challenges: Scalable Coordination in Sensor Networks," *MobiCom'99*, Seattle, WA, pp.263-270, Aug. 1999.

[3] Rabiner, W., Chandrakasan, A., Balakrishnan, H., "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Hawaii International Conference on System Sciences*, Maui, HI, pp.10-19, Jan. 2000.

[4] Chang, J.-H., Tassiulas, L., "Energy Conserving Routing in Wireless Ad-Hoc Networks.," *INFOCOM'00*, Tel Aviv, Israel, pp.22-31, Mar. 2000.

[5] Stojmenovic, I., Lin, X., "Power-Aware Localized Routing in Wireless Networks," *Proceedings IPDPS 2000*, Cancun, Mexico, pp. 371-376, May 2000.

[6] Broch, J., Maltz, D., Johnson, D., Hu, Y. Jetcheva, J., "A performance comparison of multi-hop wireless ad-hoc network routing protocols," *Mobicom'98*, Dallas, Texas, pp.85-97, Oct. 1998.

[7] Pearlman, M., Haas, Z., "Improving the Performance of Query-Based Routing Protocols through 'Diversity-Injection'," *WCNC'99*, New Orleans, LA, pp. 1546-1550, Sept. 1999.

[8] Rabiner, W., Kulik, J., Balakrishnan, H., "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *MobiCom'99*, Seattle, WA, pp. 174-185, Aug. 1999.

[9] Dattatreya, G.R., Kulkarni, S.S., "Simulation of Adaptive Statistically Multiplexed Routing in Ad Hoc Networks," *WCNC'99*, New Orleans, LA, pp. 931-935, Sept. 1999.

Appendix F:

Distributed Assignment of Encoded MAC Addresses in Wireless Sensor Networks

***Abstract*-- Technological advances have spurred the development of ad-hoc networks comprised of numerous wireless sensor nodes. The vast majority of data that needs to transverse the sensor network consists of only a few bytes, including all network and application layer identifiers. Therefore, MAC addresses, which are vital in a shared medium, present too much overhead, particularly because they are traditionally unique fixed length IDs. To tackle this overhead, we propose a new dynamic MAC addressing scheme for ad-hoc sensor nets that exploits spatial reuse. A distributed algorithm that can tolerate unidirectional links, assigns the addresses. In addition, we present a variable length address representation, based on prefix coding. Our dynamic addressing scheme scales almost perfectly with the network size due to its distributed assignment algorithm and the variable length address representation, rendering it well suited for sensor networks with thousands or millions of nodes.**

I. INTRODUCTION

Propelled by the trend towards miniaturization, wireless sensor nodes have been developed, equipped with integrated sensing capabilities, signal processing and radio communications [1][2][3]. Thousands or even millions of these nodes are **networked together in an ad-hoc fashion**, sense their surroundings and coordinate amongst each other through short-range low-power wireless transceivers. These nodes could be dropped on a battlefield or other inhospitable terrain and form an ad-hoc surveillance network [2][4]. Others envision networked sensors to monitor factory and office conditions or even wildlife [1][3]. It is well recognized that most sensor networks (like the above examples) consist out of **immobile nodes** [2][4][5][6]. Although it is possible to design networks with mobile sensors, we explicitly focus on the dominant subclass of node immobile sensor net. Because of their required unobtrusiveness for most applications, the sensor nodes have a small form-factor and rely solely on battery energy. As it is typically impossible to replace the batteries in the operating scenarios described before, low **energy consumption is critical** [2][3][4][5][6]. When networking these nodes together, the effects of this requirement ripple through the entire system, from data generation to communications and protocols. Radio communications, for example, have to be restricted in range and therefore are **multi-hop** in nature [2][6].

A crucial issue regarding energy efficiency is the observation that every bit transmitted takes a bite out of the node's lifetime [6][9]. We will show in section II **that MAC addresses contribute considerably to the header overhead** in data packets. In this

paper, we focus on the issue of reducing the size of MAC addresses in wireless sensor networks, translating directly in energy savings for data transmissions. Our distributed addressing scheme, based on **spatial reuse and encoded address representation**, reduces the MAC address size by a factor 3 to 6. Furthermore, it scales extremely well with the network size, and as such is resilient to varying or unknown network sizes and densities.

II. THE USE OF ADDRESSES

1. MAC and Network Address

Before delving into our addressing scheme, we first turn the reader's attention to the role of MAC addresses. The distinction between MAC and network layer addresses is well understood for networks such as the Internet. However, in wireless ad-hoc networks this distinction is not as crisp and often both types of addresses are condensed into one unique identifier that is used by routing, link layer and application layer protocols alike [18][6]. Consider the example in figure 1a where a packet needs to be routed from *A* to *G* via the multi-hop path *A-B-D-F-G*. The letters represent the network layer addresses of the nodes. In sensor networks, routing state is typically kept in the nodes [2][6][25]. When node *D* on the path forwards the packet to *F* it needs to include in the header both the final destination *G* and the next hop *F*. The first address, which serves as a network address, is needed for *F* to figure out the next hop to the final destination. It is clear that the network addresses need to be network wide unique to be able to identify all possible destinations. The second address, which serves the purpose of a MAC address, is needed because *B*, *C* and *E* have to know that they are not the intended receivers.

The MAC addresses, however, need not be network wide unique! We can use the MAC addresses represented by the numbers in figure 1b (the network address of each node is still the one in figure 1a), where addresses *0* and *3* have been reused. Node *D* can use MAC address *0* to identify the next-hop receiver and network address *G* for the final destination. As all neighbors of node *D* have a distinct MAC address, no confusion arises. The reason is that the functionality of the MAC address is restricted to the direct transmission neighborhood of a node, allowing careful spatial reuse. Spatial reuse leads to a decrease in number of distinct addresses used, which can therefore represented by a smaller number of bits.

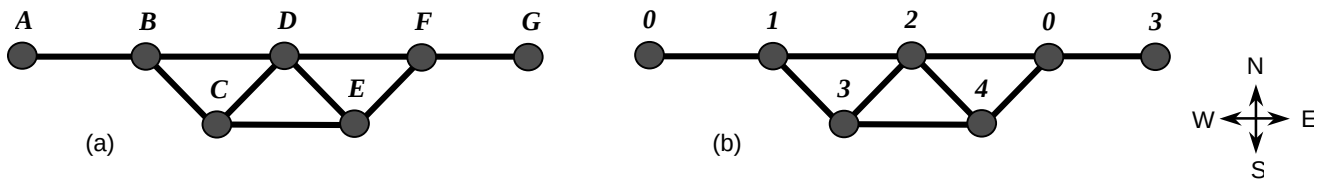


Figure 1: Decoupling address functionality, (a) network address and (b) MAC address

2. Naming and Addressing

In traditional networks, an ID based name identifies the destination. This name is then mapped onto the network address, which is used for routing. On the level of the individual hop, the MAC address identifies the next hop receiver. However, in sensor networks typical queries are not 'what is the temperature of node #27563' or 'is there a rhino near node #85396', but rather 'where is the temperature higher than 60 degrees' or 'notify me of any big animals in the south-east quadrant'. As such, **attribute based naming** schemes replace the traditional ID based ones [6][7]. Furthermore, the upfront translation from the name to the network address is omitted in favor of a lazy resolution, and the routing functionality is for a large part based directly on attributes such as location [8][11]. The reason for this approach is energy efficiency, as more common attributes can be coded in a small number of bits. Typical attributes are 'all nodes in this geographic area' (traffic to the nodes) or 'the nearest gateway' (traffic from the nodes). Each node is still equipped with a unique network address, but only very rarely will this be used for routing (exceptions are for administrative and diagnostic purposes) [2][6]. Furthermore, the majority of data traffic flows from sensor nodes to 'data sinks' [6], which typically are specialized gateway nodes that forward data to the end-users [2]. A large fraction of sensor network traffic therefore has the routing attribute 'to the nearest gateway', which can be encoded very efficiently as a single bit '0' (while the less common attributes are encoded with '1' followed by further denotation). As a result, the dominant core network traffic will carry only a **1-bit or very short name attribute** as a replacement of a network address.

3. MAC Address Overhead

Despite the attribute based naming, MAC addresses are still required in sensor networks. Assume that in figure 1 a packet needs to be routed from node *A* to the east-most side of the network. A possible routing protocol could be that a node forwards packets to all nodes that are in the target direction ±45 degrees. When node *D* forwards the packet it includes the attribute 'east' as a replacement of the network address. However, it also has to include its MAC address *2*, such that *E* knows the sender is *D* (in which case it has to forward the packet) and not *F* (in which case it would have had to drop the packet). We have assumed that each node knows the location of its neighbor through a location discovery protocol. Many other alternatives can be conceived, but they all **require the MAC address** of sender or receiver or both to be transmitted. This observation holds true not only for contention based medium access (with or without paging channel), but also for broadcast TDMA [10][11]. Also note that random identifiers [9] cannot serve the purpose of MAC addresses, as they do not guarantee the absence of collisions.

Furthermore, it has been recognized that forwarding the raw sensor data to the end-user is not a viable solution [2]. In order to conserve network resources, nodes process data locally, coordinate amongst neighbors and forward only the aggregated data or decision information [4]. Consequently, the amount of data involved in network scale communication is typically in the order of **8 to 16 bits per packet** [2][9]. Considering the extremely small amount of application data and the compact attribute names, **the**

overhead of the unique node ID as a MAC address would be prohibitive. For a wireless sensor network of 10,000 nodes, this would mean that a network wide unique MAC address requires 14 bits, which is typically around half the total packet size. A globally unique MAC address (such as an Ethernet address) would be even more overkill. As every bit transmitted reduces the lifetime of a sensor node, it is crucial to limit the packet size to an absolute minimum [9] and the goal of our research is to reduce this MAC overhead. As we have argued in figure 1, MAC addresses only need to be **unique in the transmission neighborhood of a node** and can be spatially reused. In the remainder of this paper, we will use the term 'address' to denote the MAC address.

4. Related Work

Other researchers have explored the idea of spatially reusing addresses. The scheme in [12] dynamically assigns addresses in a cellular LAN scenario. Our work does not rely on a centralized controller, which makes it more scalable and robust in terms of node failures. Related, although not identical, assignment problems have been studied for TDMA, FDMA and CDMA. In [13] a unified framework is presented that encompasses all assignment problems, including ours, but the specific constraints of address reuse are not discussed. Algorithms to solve the distributed assignment for TDMA [14][15] or CDMA [16][17] can be extended for address assignment, although they do not consider unidirectional links (which is an important issue). Some of them also assume that the network has a stationary topology at the start of the protocol, which is not the case in sensor networks. The most important difference with all the prior work on assignment algorithms is that added benefit can be found in encoding the assignment, which leverages the fact that not all addresses are used equally often. There is no analogy to this in TDMA, FDMA or CDMA. The target in address assignment is therefore not simply using as few addresses as possible, but also reuse the same ones as often as possible. We will explain this in more detail in section V.

III. ADDRESS REUSE CONSTRAINTS

In this section, we explore the constraints that govern the address assignment. It is assumed each node has a (network wide) unique ID incorporated in it and is equipped with a low-power transceiver with a bounded transmission range, typically in the order of a few tens of meters [10][11][18][19]. Furthermore, this transmission range of the radios in the different nodes may not be exactly the same due to variations in the physical device implementation as well as in the wireless propagation environment. As a consequence, communication links between two nodes are not necessarily bi-directional. Any algorithm we devise must be able to cope with both **unidirectional and bi-directional links** in order to be useful in real life scenarios.

In addition, we opt for a distributed address assignment algorithm as the network wide communication needed for a centralized algorithm comes at an extensive overall energy cost, especially when the number of sensor nodes is large. Moreover, the network topology is not perfectly constant. During deployment, the nodes typically boot independently and can remain inactive for a

substantial amount of time. Furthermore, they can fail when they run out of battery or are physically destroyed. The network has to remain operational during this entire period. It is therefore imperative that the address assignment algorithm quickly acquires a valid solution that tracks the topology changes. We call this property ***additive convergence***. A centralized algorithm would require global updates, which are too costly. Only a **distributed algorithm** is a viable option.

Figure 2 depicts a node *A* and all its neighbors. We call nodes that can both receive from and send to *A* bi-directional neighbors (in this case nodes *B* and *C*). Nodes that can only receive from *A* are called out-neighbors (*D* and *E*), while those that can only send to *A* are called in-neighbors (*G* and *F*). When a node has a packet to transmit, each neighbor has to be able to figure out whether or not it is the intended receiver. Intuitively, this means that all bi-directional and out-neighbors need to have a different address. However, in a distributed algorithm this condition is impractical. From figure 2, it is clear that node *A* has no direct means of gathering information on the addresses that *D* and *E* choose. It does not even know of their existence!

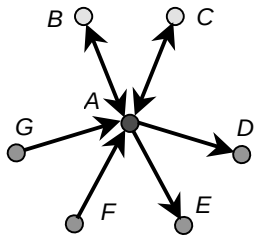


Figure 2: Address constraints for bi- and unidirectional links

We can tackle this issue by restricting the data communication to the bi-directional links only. In this case, *A* can only forward data to *B* or *C*. This restriction makes sense for typical higher layer protocols that rely on acknowledgments, path reversals or otherwise assume bi-directional links. The intended receiving nodes, *B* and *C*, need distinct addresses of course. Nodes *D* and *E* still overhear the transmission, but they are never the intended receiver. Of course, both *D* and *E* need to be able to identify the link as being one of their unidirectional links, or equivalently whether this transmission is from one of their in-neighbors or from one of their bi-directional neighbors. When the sender address is included in the packet as well, the fact that they hear the transmission completely identifies the link. Knowing that they only have a unidirectional link to *A*, they can conclude they are not the intended receivers. We assume that each node has established its bi-directional neighbors through a discovery protocol prior to the address assignment. The resulting constraints are given by lemma 1. They specify a valid address assignment that can be achieved by a distributed algorithm in the presence of unidirectional links. These constraints are new to the best of our knowledge and therefore define a new assignment problem.

Lemma 1: *When the normal-mode data communication of node A is restricted to its bi-directional neighbors, a valid assignment of addresses is such that all bi-directional neighbors have distinct addresses and that all in-neighbors have addresses different from those of the bi-directional neighbors, for any node A.*

The first part of lemma 1 ensures that *B* and *C* have different addresses, since they can be intended receivers of node *A*. The second part of this lemma tells that *F* and *G* have addresses different from those of *B* and *C*, although the addresses of *F* and *G* themselves do not need to be different. As a consequence, node *A* can distinguish between its senders with which it has a unidirectional link and with which it has a bi-directional link. Although node *A* cannot send information back to *F* or *G*, it can instruct *B* and *C* to choose addresses that do not conflict with those of *F* or *G*.

IV. DISTRIBUTED ASSIGNMENT ALGORITHM

1. Algorithm Description

The pseudo-code of our distributed assignment algorithm is presented in figure 3. Its core operation is based on a periodic broadcast packet: ***broadcast_pkt*** (line 7). This packet contains the node's neighborhood information: its own address and those of its neighbors it is has knowledge of (i.e. bi-directional and in-neighbors). The addresses of these neighbors are obtained by listening to their periodic broadcasts. Each node therefore obtains information on its one-hop and two-hop neighbors, which is stored in the structure ***constraints*** (line 10). This information is "soft-state" and a timeout invalidates the entry for a particular neighbor if that node is not heard from for a while (line 6). Going back to figure 2, node *B* can learn about the addresses of *C*, *F*, and *G* by listening to the periodic broadcasts of *A*. One cycle after a node boots up, it chooses an address that satisfies lemma 1 based on the information in ***constraints*** (line 5). A detailed description of how valid address is chosen will be discussed in subsection V.2.

Since periodic transmissions are also needed for connectivity discovery/updating and for other network maintenance and management protocols, the neighborhood information could be piggybacked onto them with only a few bits added [2]. Furthermore, the period of the broadcast (which we call the *cycle time*) can be increased once the transient boot-up phase is over and made comparable to the expected time constant of the network dynamics (i.e. frequency of node failures, etc.). As an alternative to the (adaptive) periodic broadcasts, we can opt for a reactive scheme with explicit *request()* packets to solicit a ***broadcast_pkt*** from neighboring nodes. Now, the nodes send out this request when entering the network, or when suspecting changes in topology due to nodes failures (through indications from higher layers). Which of these options is the best depends on the specific network dynamics and deployment scenario. In any case, the expected number of ***broadcast_pkt*** over the lifetime of a node is rather small. We will discuss the tradeoff between the protocol overhead and the savings in address size in section V.

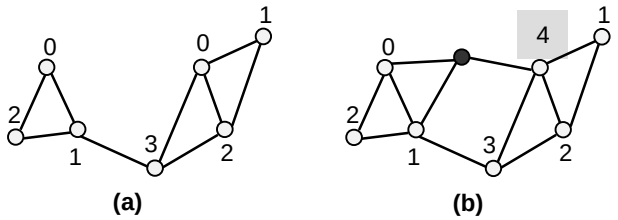
```
 1 set(timer);
 2 while (1): wait for event
 3  case event == timer
 4    if (no_addr())
 5      choose_addr(constraints);
 6    check_timeout(constraints);
 7    send(broadcast_pkt);
 8    set(timer);
 9  case event == broadcast_pkt
10    update(constraints);
11    if (addr_conflict(constraints))
12       send(conflict_pkt);
13  case event == conflict_pkt
14    update(constraints);
15    choose_addr(constraints);
16    send(broadcast_pkt);

17 end while
```

Figure 3: Distributed address assignment algorithm

At each instant in time, the network consisting of the active nodes with an address has a valid assignment and is therefore operational. This algorithm therefore satisfies the property of *additive convergence*. The final address selection corresponds to the centralized scheme with random ordering presented in [12]. However, the additive convergence property also requires extra provisions in the algorithm. Even though the active nodes have chosen a valid, i.e. non-conflicting, address, the waking up of a new node, may invalidate the existing address assignment as illustrated in figure 4.

At first, all active nodes have chosen a valid address based on the algorithm we have just explained (figure 4a). When the dark node boots up (figure 4b), its mere presence causes a new address conflict between the nodes with address *0*. The new node detects this problem and instructs one of these two nodes to choose another address (in this example, address 4). Lines 11 to 16 in the algorithm are devoted to the detection and resolution of such problems. When a node, in the above example the dark node, detects an address conflict while receiving a ***broadcast_pkt***, it orders one of the nodes with a bi-directional link to choose another address by sending a ***conflict_pkt*** packet. If the conflict is simply between in-neighbors, no action is taken since it is both not necessary (see lemma 1) and impossible to correct. When receiving this ***conflict_pkt*** packet, a node chooses a new address the same way it did before, now based on updated ***constraints*** information. A new ***broadcast_pkt*** is transmitted to inform the other nodes that a new address was chosen and the old one is freed up.



(a) (b)

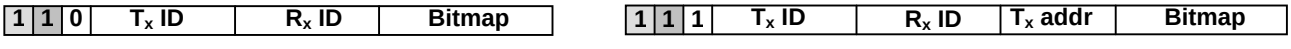
Figure 4: Address conflicts due to a new node

2. Packet Formats

Finally, we present the packet formats we propose to use. Figure 5a illustrates the structure of a ***broadcast_pkt***. The first bit identifies the packet as being a control message of our protocol. The second bit marks it as a periodic broadcast message, thereby fixing the type of fields that follow. The next two fields contain the unique node ID and address of the sender. The 'Bitmap' encodes the addresses of the sender's neighbors. Although we will describe how a specific address is chosen amongst the valid ones in the section V, for now it suffices to know that a node selects the lowest address that does not collide. As a result, the addresses of the neighbors are more likely to be towards the lower end of the address range. The bitmap field in figure 5b exploits this property, where a '1' at position *x* indicates that a neighbor has picked this address. The lower L_b addresses are encoded this way. They are followed by a single bit indicating whether another set of L_b addresses is appended ('1') or not ('0'). We choose L_b equal to 16 in our implementation. Figure 6 illustrates the structure of the conflict message. Again, the first two bits identify the packet as being a control packet and a conflict message respectively. The third bit is set to '0' if the packet does not contain the address of the sender (figure 6a), which occurs when it has not chosen an address yet. Since these messages are sent to a particular receiver, the ID of this receiver is needed as well.

(a) (b)

Figure 5: (a) Broadcast message and (b) bitmap field



(a) (b)

Figure 6: Conflict message

V. ADDRESS SELECTION AND REPRESENTATION

1. Address Representation

Thus far we have only specified how a node learns about the addresses it may not pick, but we have not detailed yet how it chooses an address from the remaining ones. In fact, this choice is tightly linked to the way we propose to represent the addresses in data packets. Therefore, we discuss our address representation first. For now, it suffices to observe that the number of addresses needed is related to the number of neighbors of a node. This directly follows from the formulation of lemma 1.

Traditionally, protocols represent addresses using a fixed length address field, such as in Ethernet or IP. We denote the total number of sensor nodes in the area under scrutiny by ***N***. The algorithm we have presented permits reuse of addresses and therefore reduces the maximum address needed from *N* to a smaller number. The highest address any node in the sensor field ends up with, clearly determines the number of bits needed. This typically corresponds to a node that has a large number of neighbors, or alternatively where locally the network density is high. Even when we know the approximate network density, it is hard to predict the maximum address needed. Global communication to determine the maximum address is undesirable. Therefore, enough **safety margin** needs to be provided to account for rare but not impossible cases of high local densities. When the average network density is not known a priori, the situation is even more unfavorable.

We propose a new alternative way of representing an address. Our scheme, which we call ***encoded address representation***, has the advantage that it typically requires less bits then the fixed representation. Furthermore, it exhibits graceful scaling properties with network density and is perfectly independent of network size (i.e. scales perfectly with *N*). Our encoded scheme does not transmit the address itself, but a codeword representing the actual address. The codeword itself is prefix coded, which means that the end can be identified as it is encountered [20]. Although the **length of the codeword is variable**, it can uniquely be deduced from the codeword itself. More specifically, we utilize the well-known Huffman coding. When for each address the probability of occurrence is known, this scheme results in a minimum average codeword length. In subsequent data packets, instead of sending the address, we use the codeword instead. The address selection protocol uses regular integer numbers as addresses. However, in data transmissions, the codewords can perfectly assume the role of addresses.

2. Address Selection

From the previous discussion, it is apparent that it is beneficial to reuse addresses as often as possible, such that their probability of occurrence is higher. Although it does not really matter which address is used the most, from a practical standpoint it is easiest to try to reuse the lower addresses as frequently as possible. This corresponds perfectly to intuition. Next, we illustrate the address selection for an **example** setup with a specific node density. In section VI, we will discuss how the performance

depends on the density and other operation parameters. In figure 7, the continuous solid curve, labeled '*incremental*', corresponds to the algorithm of figure 3 where the lowest non-conflicting address is chosen in the function `choose_addr()`. It illustrates the frequency at which each address was eventually assigned. These results were obtained through simulations written on the Parsec platform, an event-driven parallel simulation language [21]. We assume *N* nodes are distributed randomly over a square field of size *L* x *L* and each of them has a transmission range *R*. For a uniform network density, the average connectivity depends only on the average number of neighbors of a node, denoted by parameter . .

$$. = \frac{N}{L^2} \bullet \pi R^2 \tag{1}$$

Since the address assignment depends solely on the network connectivity, it is logical that its performance depends only on . and not on *N*, *R* and *L* separately. We have verified this statement through simulations. In the example of figure 7, *N* = 500, L^2 = 50,000 m^2 and *R* = 17.84 m (which is a reasonable value for sensor nets [18]). This results in . = 10. A node boots at a random time in an interval of 10 seconds. The initial broadcast cycle is set to 10 seconds. Once the network has booted up, the period can be extended or we can switch to a reactive scheme, without any noticeable influence on the performance. The results are averaged over 500 simulation runs.

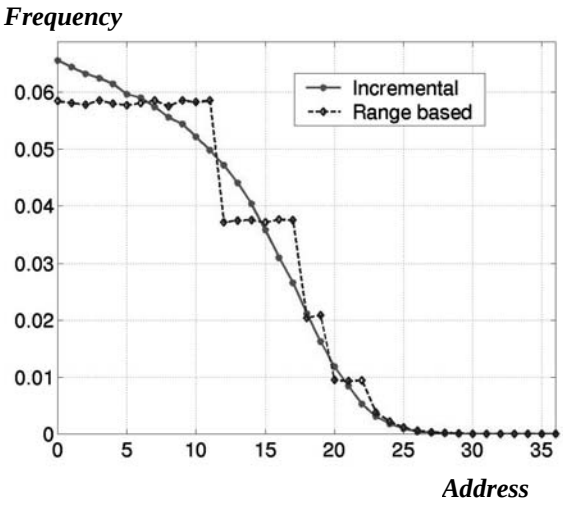


Figure 7: Example of address selection frequency

Based on this 'incremental' curve, the optimal encoded address format can be derived through Huffman coding. The results are listed in table 1. For addresses higher than 23, the codeword can be obtained from the codeword of the previous address by replacing the ending ***0*** by ***10***. The average encoded address, which is calculated by multiplying the address size with its frequency of occurrence, requires only **4.41** bits. The fixed addressing scheme would have resulted in at least 6 bits, since the maximum address encountered was 34.

Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13
# bits	4	4	4	4	4	4	4	4	4	4	4	4	5	5
Code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100:0	1100:1

Address	14	15	16	17	18	19	20	21	22	23
# bits	5	5	5	5	6	6	7	7	7	8
Code	1101:0	1101:1	1110:0	1110:1	1111:0:0	1111:0:1	1111:1:00	1111:1:01	1111:1:10	1111:1:11:0

Table 1: Example of address encoding corresponding to figure 7

Upon further inspection of table 1, we notice that several addresses require the same number of code bits, what we define as being in the same ***range*** (e.g. addresses *0* to *11* are encoded in 4 bits). From a practical perspective, all of these addresses are equally efficient. Instead of choosing the lowest non-conflicting address in `choose_addr()`, selecting a random address in the lowest range possible results in the same average code length. The dashed curve (called 'range based') in figure 7 illustrates this alternative. We see indeed that all addresses that result in the same number of bits are selected equally often. The average dynamic address size is still 4.41 bits.

Furthermore, our simulations confirm the intuition that the 'range based' curve can be derived from the 'incremental' one, by averaging the probability of all addresses in the respective range. For example, the probability of the first twelve addresses for the 'range based' scheme is equal to one twelfth of the sum of the probabilities of addresses *0* to *11* in the incremental scheme. Even though in practice addresses are picked randomly in a range, we can thus perfectly analyze the performance in number of address bits needed by investigating the incremental scheme. In the remainder of this paper, we therefore focus on the distribution of addresses chosen and **need to consider only the incremental case** since it encompasses the range based one as well.

3. Algorithm Overhead

Although the average address size is the same for the incremental and range based address selection, there is a clear advantage in control overhead as illustrated in table 2. This table lists for both curves of figure 7 the average number of packets and bits sent per node in the `send()` statements of lines 12 and 16. We do not include the periodic broadcasts, since they add a fixed overhead, which depends on the period and on the potential for piggybacking. For this example, the encoding of the **addr** field of the control messages (see figures 5 and 6) was according to table 1. The **ID** field, on the other hand, depends on the anticipated network size. In our simulations, we allocated 14 bits.

	conflict_pkt		*broadcast_pkt*	
	Packets	Bits	Packets	Bits
Incremental	0.546	25.34	0.546	16.19
Range based	0.319	18.66	0.319	12.11

Table 2: Control overhead per node

We can use this information to evaluate the overall tradeoff between protocol overhead and savings in address size. The protocol overhead expressed in number of bits is denoted as B_O. From table 2, we can derive that for the range-based scheme, B_O is given by (2). In this expression, M is the number of periodic broadcasts over the entire lifetime of a node (the packet size is derived from our simulations, which incorporate piggybacking). We also know that the number of bits saved B_S is equal to (3), compared to a network wide unique MAC address of 14 bits. Here, P is the total number of data packets sent by a node and the factor 2 accounts for the fact that both the sender and receiver MAC address is incorporated in the packets.

$$B_O = (18.66 + 12.11) + M \bullet (23.95) \tag{2}$$

$$B_S = 2 \bullet (14 - 4.41) \bullet P \tag{3}$$

Our dynamic addressing scheme results in an overall decrease in number of bits sent if $B_O < B_S$ or equivalently when:

$$P > 1.2 \bullet M + 1.6 \tag{4}$$

Although some nodes might not send any traffic at all, e.g. when there is absolutely no activity to report in their sector, these nodes are not critical for the overall network lifetime anyway. It is the nodes that forward a lot of traffic that are the critical ones, and for these (4) is likely to be satisfied. Further note that only the initial cycle is short. Once the cycle is matched to the network dynamics or is switched to the reactive mode, the number of broadcast packets is limited. The exact value of M strongly depends on the network deployment and sensing scenario, as node failures and reactivation follow directly from the network traffic patterns, routing strategies and overall network management. Our simulations show that M is typically to the order of 20-50. We therefore expect that condition (4) will be satisfied for almost all sensor networks.

We would like to emphasize that the numerical results of figure 7, table 2 and the analysis above are bound to the settings in this particular example. As we will illustrate in section VI, other network densities result in different values. Since the address encoding of table 1 is derived from figure 7, it is thus optimized for these particular settings. Other network densities will have another optimal address encoding. For practical applications, the network density will have to be estimated, such that a good encoding can be selected. We will revisit this issue in section VI.

4. Computational requirements

A possible drawback of our encoded address scheme is the fact that codewords instead of addresses are transmitted. During data transmission, the sender needs to know the codeword of the intended receiver, which can be stored in a table. At the receiver side, some more work is needed since the encoded addresses do not have a fixed length. In fact, the codewords have to be stepped through bit by bit in order to detect their end. However, this overhead in header processing is negligible compared to the savings in transmission energy.

Communication	
RFM 2.4 kbps over 20 m	0.18 ∝J / bit
RFM 2.4 kbps over 80 m	0.94 ∝J / bit
Conexant RDSSS9M 100 kbps over 100 m	1 ∝J / bit
Computation	
StrongARM SA-1100 (150 MIPS, 1.5 V)	1.5 nJ/instr
StrongARM SA-1100 (250 MIPS, 2.0 V)	2.2 nJ/instr

Table 3: Communication and computation costs

Table 3 illustrates this claim by comparing the communication cost (with the low-power RFM [22] or Conexant [18] radio) versus the computation cost (on a low-power StrongARM SA-1100 embedded processor [23]). For these devices, which are believed representative for sensor network and are used by various research groups working in this field [18][19], the cost of transmitting 1 bit is equivalent to 80 to 600 instructions. The savings therefore clearly outweigh the cost of Huffman decoding (which should not take more than a few tens of instructions on average).

VI. PROTOCOL ANALYSIS

1. Influence of Network Density

To evaluate the influence of network density, figure 8 depicts the performance of our algorithm for different values of . assuming a uniform node density. It is apparent that the less connected the network is (lower .), the more frequent the lower addresses can be used. This is intuitively clear as fewer neighbors cause less address conflicts. We have also evaluated the memory requirements of our algorithm for different values of . . Since the required amount of storage in a node depends on its number of neighbors, which varies between nodes, we pick a dynamic memory allocation scheme. Table 4 lists the required storage for both the average and worst case (for N = 500 nodes, but these values are almost independent of N).

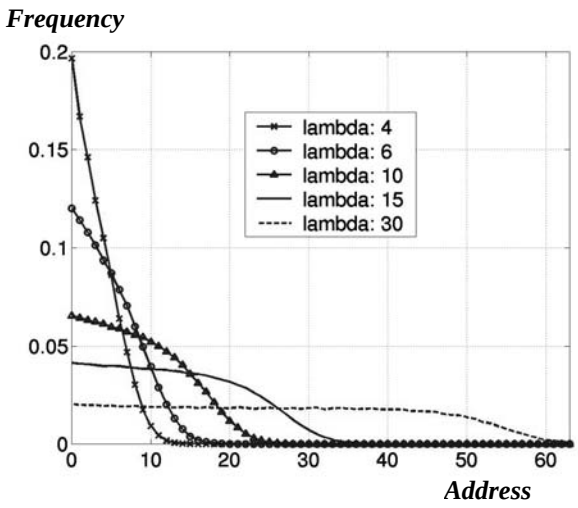
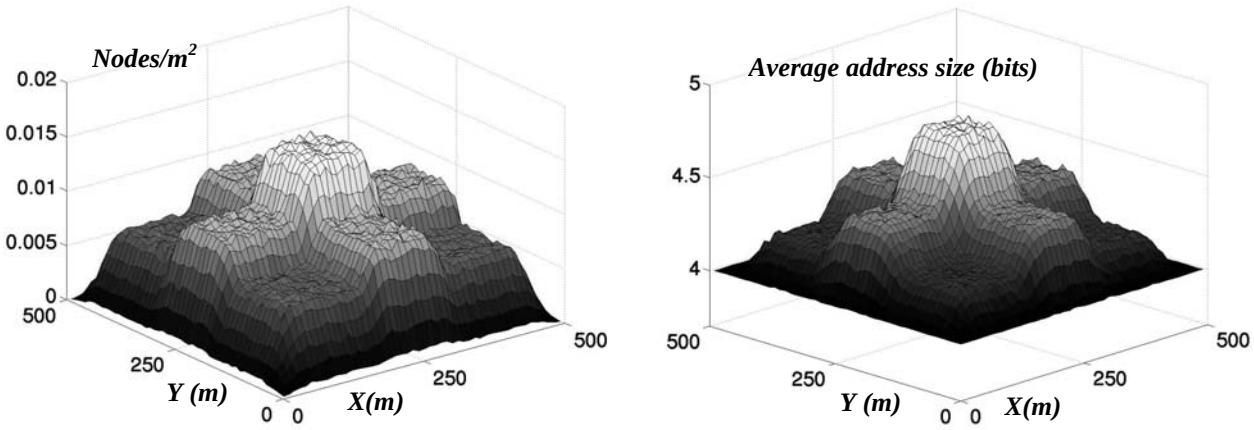


Figure 8: Addresses chosen for different value of .

.	5	10	15	20
Average (bytes)	23	47	71	91
Worst case (bytes)	68	142	217	293

Table 4: Average memory requirements per node

Up until now, we have always assumed a uniform node density. To illustrate that our analysis does not depend on this assumption, we have simulated the algorithm on a field with the non-uniform density, where nodes are randomly distributed according to figure 9a. The second highest level (the arms of the cross shape) corresponds to . = 10. The spatial distribution of the address size, averaged over 1,500 simulation runs, is given in figure 9b. We used the address encoding of table 1.



(a) **(b)**

Figure 9: Non-uniform network density

We observe that the average address size at the level with $\bar{\rho}$ = 10 corresponds to 4.41 as in the uniform density case. For the other regions, we can look at the value of $\bar{\rho}$ in the locality of a node and pick the corresponding curve from figure 8. The average occurrence frequency of each address together with the codeword size of that address (given in table 1) allows us to predict the performance of figure 9 in these other regions as well. This and other simulations we ran, show that the address size depends only on the density in "the locality of a node", corresponding roughly to circle with radius 3*R*.

Remember that the address representation of table 1 optimized for $\bar{\rho}$ = 10. In the other regions with higher or lower values, this representation is no longer optimal. A great benefit of our encoded scheme is that it is **scalable**, such that all addresses can be represented no matter what. However, in the case of the fixed size scheme, the address field would be selected based on the expected maximum value of $\bar{\rho}$, with some margin. If locally the density is higher, this size can turn out to be insufficient to represent all addresses, causing major problems.

How can our encoded dynamic addressing scheme be used **in practice**? Typical scenarios are likely to exhibit non-uniform network densities, which are sufficiently smooth (i.e. without abrupt transitions). If we are able to estimate the expected node densities before the network deployment, we can predict the behavior of any address encoding choice based on the curves of figure 8, as explained before. By taking a weighted average, we can choose the optimal encoding. If we do not have such reasonable estimate on the node densities, we simply guess a good encoding scheme. Any encoding, optimal or not, is always able to represent all possible addresses. Before deployment of the network, the **encoding scheme is hard-coded into all the nodes**, as it needs to be network wide consistent. This option is most likely superior to executing a network wide "consensus" algorithm after the address selection phase.

2. Random Packet Losses

We analyze the effects of random packet losses by assuming that each packet has a probability P_{drop} of not reaching a receiving node. This procedure models losses due to MAC congestion, problems in the wireless link (fading, shadowing, noise, jamming) or any other source.

Figure 10a depicts simulation results for different loss probabilities. It is clear that packet losses up to 10% have almost no effect on the performance of our algorithm in terms of final addresses chosen. It is therefore very resilient against these degradations for typical operating conditions. Another aspect worth investigating is the convergence time of our scheme, i.e. how long it takes until every node has a valid address. Simulations show that this aspect is independent of all the parameters discussed so far. Specifically, convergence occurs one cycle after the last node is created (in fact, each node acquires a valid address one cycle after its creation). However, this changes when random packet losses are included. Figure 10b plots the interval between the time the last node boots and the time that all nodes have acquired a valid address.

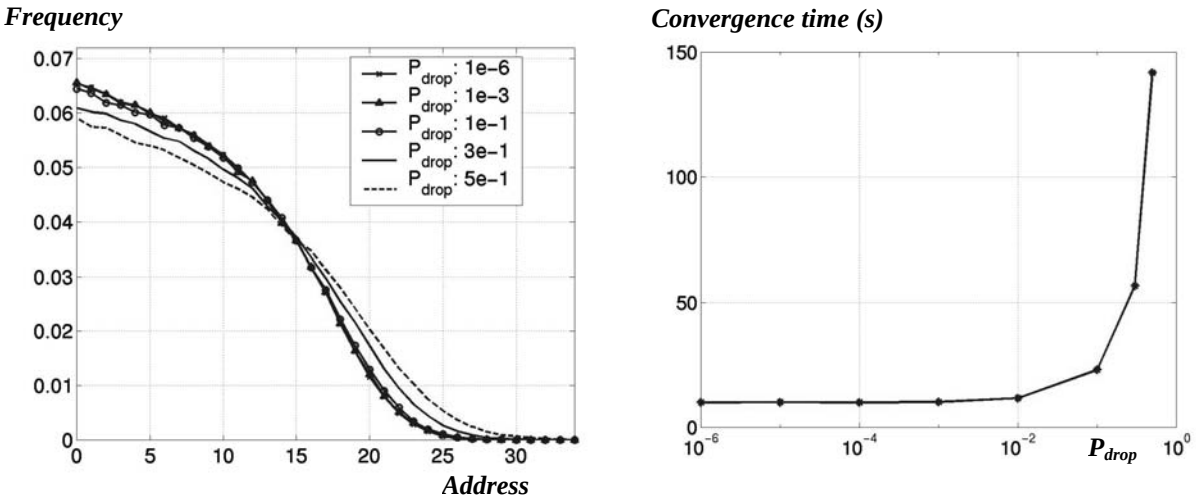


Figure 10: Effects of packet losses (. = 10)

3. Unidirectionality

As explained in section III, our algorithm is specifically designed to accommodate both unidirectional and bi-directional links. All simulations up until now only included bi-directional links. Figure 11 illustrates the impact of having unidirectional links. The transmission range *R* of each node is now chosen uniformly random in an interval [*R* - *W*/2 , *R* + *W*/2]. By increasing *W*, the fraction of unidirectional links increases (from 0% to around 50% for *W* from 0 to 20). We observe that the performance degrades with increasing *W*. For all our simulations, we have also verified that the assigned addresses indeed satisfy lemma 1.

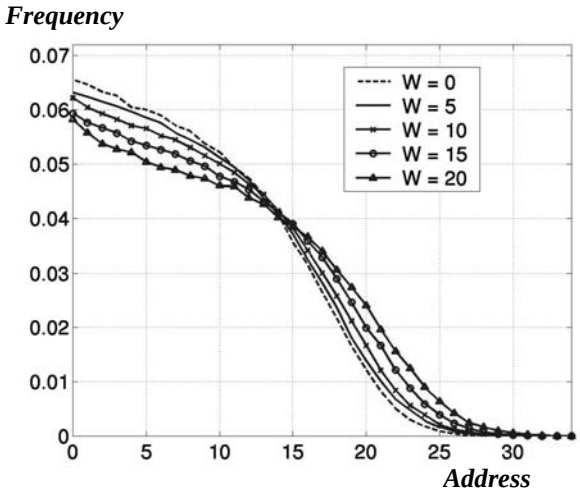


Figure 11: Performance with bi- and unidirectional links (. = 10)

VII. COMPARISON BETWEEN ADDRESSING SCHEMES

1. Scalability of Address Representations

The performance of our encoded address scheme depends on curves such as those of figure 8. Our simulations (which are not included here due to space limitations) show that these curves are virtually independent of the network size. The reason is the fact that the address assignment is basically dictated by the connectivity in the locality of the nodes. However, edge effects change this statement slightly. Our simulations show, however, that the **encoded address representation scales perfectly** with *N* when these edge effects are disregarded. The representation with a **fixed size address field does not scale well**. This is due to the fact that when the network size increases, it is more likely that at least one node has a higher degree (and thus needs at least that many addresses for its neighbors).

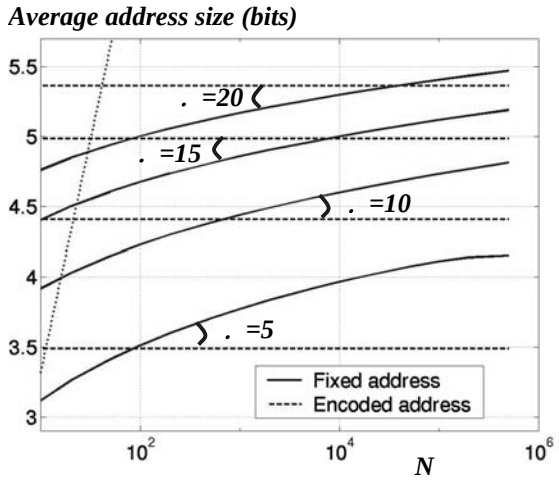


Figure 12: Comparison between fixed and encoded address representation

Figure 12 compares the performance of the two address representations. Both reuse addresses spatially, but the way they are represented differs. It is clear that the encoded scheme has superior scaling properties. Furthermore, for most of the network sizes of interest, the **encoded scheme is preferable to the fixed size one**. The steep dotted curve represents the address size that would be needed for a network-wide unique ID, which is clearly undesirable.

2. Overall Comparison

When comparing MAC addressing alternatives for sensor networks, **scalability** is a key issue, which consists of two elements:

1. When the assignment of addresses is static (i.e. before deployment), there is no true scaling issue. Dynamic address **assignment protocols**, on the other hand, can be centralized or distributed, where only distributed ones scale well.

2. The second aspect is **address size**. Without spatial reuse, scaling is very poor. On the other hand, scaling is perfect when encoding spatially reused addresses (see section VII.1). This is not true anymore when representing these addresses in a fixed size field.

Table 5 compares different approaches for N = 10,000 and $\bar{d}$ = 10. The 'address selection type' and 'address size scalability' reflect the two elements of scaling respectively. A globally unique ID would be prohibitively long (but independent of the network size, thus having perfect 'address size scalability'). Alternatively, a network wide unique ID can be allocated when the network is deployed. This option scales only logarithmically with the network size. Our first new scheme, called ***fixed size dynamic***, reuses addresses spatially, but still represents them in a fixed size field. It does not exhibit perfect address size scalability. Addresses can be assigned with a centralized or a distributed algorithm, where the last one is probably preferable. Our second scheme, called ***encoded dynamic***, achieves perfect scaling by address encoding and its distributed nature (although in theory, a centralized algorithm is possible). It also results in smaller addresses for typical network setups.

	Scheme	Address selection type	Av. size (bits)	Address size scalability
	Globally unique	Manufacturing	128	**+**
	Network wide unique	Deployment	14	–
Our schemes	Fixed size dynamic	Centr. / Distr.	**4.7**	±
Our schemes	**Encoded dynamic**	**Distributed**	**4.4**	**+**

Table 5: Comparison of addressing schemes

VIII. CONCLUSIONS

In sensor networks, the dominant traffic typically consists of packets with a small payload and a short destination name attribute. In this case, MAC addresses contribute a considerable packet header overhead. However, they cannot be simply omitted, as their functionality is needed to discern intended from non-intended receivers. We have tackled to the problem of limiting this MAC address overhead, thereby reducing the number of bits that need to be transmitted. As every bit saved relaxes the demands on the node's energy resources, this scheme eventually targets increasing the network operation lifetime. We propose a dynamic addressing scheme, which is viable when the network dynamism is low. This is the case for sensor networks with immobile nodes, but where node failures can occur. In this paper, we have presented an address assignment protocol that

1. is fully distributed,
2. spatially reuses addresses,
3. encodes the addresses using prefix codes.

Our scheme scales very well with the size of the network, because of both the distributed nature of the algorithm (which relies only on local message exchanges) and the address encoding. Both the scalability and small average address size make our scheme compare favorably to other options. We have also illustrated that the protocol is robust in the presence of packet losses and unidirectional links.

In our future work, we will look at the relationship between encoded addressing and header compression [24]. Header compression has thus far been considered only for point-to-point links, but to be useful for MAC addresses we need to take into account the inherent broadcast functionality. In this case, loosing synchronization is compounded between the different receivers, thereby limiting the compression gain. In any case, header compression is orthogonal to our scheme and their interaction seems an interesting topic for future research.

REFERENCES

[1] Ward, A., Jones, A., Hopper, A., "A new location technique for the active office," *IEEE Personal Communications* Magazine, Vol.4, No.5, pp. 42-47, October 1997.

[2] Sohrabi, K., Gao, J., Ailawadhi, V., Pottie, G., "Protocols for self-organization of a wireless sensor network," IEEE Personal Communications Magazine, Vol.7, No.5, pp. 16-27, Oct. 2000.

[3] Saffo, P., "Sensors: the next wave of innovation," Communications of the ACM, Vol.40, No.2, pp. 92-97, Feb. 1997.

[4] Pottie, G., "Hierarchical information processing in distributed Sensor networks," Proceedings of ISIT'98 Conference, Cambridge, MA, pp. 163, August 1998.

[5] Rabiner Heinzelman, W., Kulik, J., Balakrishnan, H., "Adaptive protocols for information dissemination in wireless sensor networks," Proceedings ACM/IEEE MobiCom'99, Seattle, WA, pp. 174-185, August 1999.

[6] Estrin, D., Govindan, R., "Next century challenges: scalable coordination in sensor networks," Proceedings of ACM/IEEE MobiCom'99, Seattle, WA, pp. 263-270, August 1999.

[7] Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., Lilly, J., "The design and implementation of an intentional naming system," Operating Systems Review, Vol.33, No.5, pp.186-201, December 1999.

[8] Navas, J., Imielinski, T., "GeoCast-geographic addressing and routing," Proceedings of ACM/IEEE MobiCom '97, Budapest, Hungary, pp. 66-76, September 1997.

[9] Elson, J., Estrin, D., "Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks," to appear in ICDCS'01, Phoenix, AZ, April 2001.

[10] Kahn, J., Katz, R., Pister, K., "Mobile Networking for Smart Dust", Proceedings of ACM/IEEE MobiCom'99, Seattle, WA, August, 1999.

[11] Kumar, S., "DARPA Sensor Information Technology (SensIT) Project," http://www.darpa.mil/ito/research/sensit/.

[12] Bharghavan, V., "A dynamic addressing scheme for wireless media access," Proceedings of IEEE ICC '95, Seattle, WA, pp. 756-760, June 1995.

[13] Ramanathan, S., "A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks," Proceedings of IEEE INFOCOM '97, Kobe, Japan, pp. 900-907, April 1997.

[14] Ephremides, A, Truong, T., “Distributed algorithm for efficient and interference-free broadcasting in radio networks,” Proceedings of IEEE INFOCOM '88, New Orleans, LA, pp. 1119-1124, March 1988.

[15] Cidon, I., Sidi, M., “Distributed assignment algorithms for multihop packet radio networks,” IEEE Transactions on Computers, Vol.38, No.10, pp. 1353-1361, October 1989.

[16] Hu, L., “Distributed Code Assignments for CDMA Packet Radio Networks,” IEEE/ACM Transactions on Networking, Vol.1, No.6, December 1993.

[17] Bertossi, A.A., Bonuccelli, M.A., “Code assignment for hidden terminal interference avoidance in multihop packet radio networks,” IEEE INFOCOM '92, Florence, Italy, pp. 701-709, May 1992.

[18] Rockwell Science Center, “Wireless Integrated Network Systems,” http://wins.rsc.rockwell.com/ .

[19] MIT, “∝AMPS Project,” http://www-mtl.mit.edu/ research/icsystems/ uamps/.

[20] Cover, T., Thomas, J., “Elements of Information Theory,” Wiley Series in Telecommunications, 1991.

[21] Bagrodia, R., Meyer, R., Takai, M., Chan, Y.A., Zeng, X., Marting, J., Song, H.Y., “Parsec: a parallel simulation environment for complex systems,” Computer, Vol.31, No.10, pp. 77-85, October 1998.

[22] “ASH Transceiver Designer’s Guide,” http://www.rfm.com.

[23] “Intel® StrongARM SA-1100 Microprocessor for Portable Applications Data Sheet,” http//www.arm.com.

[24] Giovanardi, A., Mazzini, G., Rossi, M., Zorzi, M., “Improved Header Compression for TCP/IP Over Wireless Links,” *Electronic Letters*, Vol. 36, No. 23, November 2000.

[25] Sensoria Corporation, http://www.sensoria.com/.

Energy Aware Wireless Sensor Networks

Vijay Raghunathan, Curt Schurgers, Sung Park, and Mani B. Srivastava
Department of Electrical Engineering, University of California, Los Angeles, CA 90095.

INTRODUCTION

Self-configuring wireless sensor networks can be invaluable in many civil and military applications for collecting, processing, and disseminating wide ranges of complex environmental data. They have therefore, attracted considerable research attention in the last few years. The WINS [1] and SmartDust [2] projects for instance, aim to integrate sensing, computing, and wireless communication capabilities into a small form factor to enable low-cost production of these tiny nodes in large numbers. Several other groups are investigating efficient hardware/software system architectures, signal processing algorithms, and network protocols for wireless sensor networks [3], [4], [5].

Sensor nodes are battery-driven, and hence operate on an extremely frugal energy budget. Further, they must have a lifetime on the order of months to years, since battery replacement is not an option for networks with thousands of physically embedded nodes. In some cases, these networks may be required to operate solely on energy scavenged from the environment through seismic, photovoltaic, or thermal conversion. This transforms energy consumption into the most important factor that determines sensor node lifetime.

Conventional low-power design techniques [6] and hardware architectures only provide point solutions which are insufficient for these highly energy constrained systems. Energy optimization, in the case of sensor networks, is much more complex, since it involves not only reducing the energy consumption of a single sensor node, but also maximizing the lifetime of an entire network. The network lifetime can be maximized only by incorporating energy-awareness into every stage of wireless sensor network design and operation, thus empowering the system with the ability to make dynamic tradeoffs between energy consumption, system performance, and operational fidelity. This new networking paradigm, with its extreme focus on energy efficiency, poses several system and network design challenges that need to be overcome to fully realize the potential of the wireless sensor systems.

A quite representative application in wireless sensor networks is event tracking, which has widespread use in applications such as security surveillance and wildlife habitat monitoring. Tracking involves a significant amount of collaboration between individual sensors to perform complex signal processing algorithms such as Kalman Filtering, Bayesian Data Fusion, and Coherent Beamforming. This collaborative signal processing nature of sensor networks offers significant opportunities for energy management. For example, just the decision of whether to do the collaborative signal processing at the user end-point or somewhere inside the network has significant implication on energy and lifetime. We will use tracking as the driver to illustrate many of the techniques presented in this paper.

PAPER OVERVIEW

This paper describes architectural and algorithmic approaches that designers can use to enhance the energy awareness of wireless sensor networks. The paper starts off with an analysis of the power consumption characteristics of typical sensor node architectures, and identifies the various factors that affect system lifetime. We then present a suite of techniques that perform aggressive energy optimization while targeting all stages of sensor network design, from individual nodes to the entire network. Maximizing network lifetime requires the use of a well-structured design methodology, which enables energy aware design, and operation of all aspects of the sensor network, from the underlying hardware platform, to the application software and network protocols. Adopting such a holistic approach ensures that energy awareness is incorporated not only into individual sensor nodes, but also into groups of communicating nodes, and the entire sensor network. By following an energy-aware design methodology based on techniques such as in this paper, designers can enhance network lifetime by orders of magnitude.

WHERE DOES THE POWER GO?

The first step in designing energy aware sensor systems involves analyzing the power dissipation characteristics of a wireless sensor node. Systematic power analysis of a sensor node is extremely important to identify power bottlenecks in the system, which can then be the target of aggressive optimization. We analyze two popular sensor nodes from a power consumption perspective, and discuss how decisions taken during node design can significantly impact the system energy consumption.

The system architecture of a canonical wireless sensor node is shown in Figure 1. The node is comprised of four subsystems: (i) a computing subsystem consisting of a microprocessor or microcontroller, (ii) a communication subsystem consisting of a short range radio for wireless communication, (iii) a sensing subsystem that links the node to the physical world and

consists of a group of sensors and actuators, and (iv) a power supply subsystem, which houses the battery and the DC-DC converter, and powers the rest of the node. The sensor node shown in Figure 1 is representative of commonly used node architectures such as [1], [2].

MICRO CONTROLLER UNIT (MCU)

Providing intelligence to the sensor node, the MCU is responsible for control of the sensors, and execution of communication protocols and signal processing algorithms on the gathered sensor data. Commonly used MCUs are Intel's StrongARM microprocessor and Atmel's AVR microcontroller. The power-performance characteristics of MCUs have been studied extensively, and several techniques have been proposed to estimate the power consumption of these embedded processors [7], [8]. While the choice of MCU is dictated by the required performance levels, it can also significantly impact the node's power dissipation characteristics. For example, the StrongARM microprocessor from Intel, used in high end sensor nodes, consumes around 400mW of power while executing instructions, whereas the ATmega103L AVR microcontroller from Atmel consumes only around 16.5mW, but provides much lower performance. Thus, the choice of MCU should be dictated by the application scenario, to achieve a close match between the performance level offered by the MCU, and that demanded by the application. Further, MCUs usually support various operating modes, including *Active*, *Idle*, and *Sleep* modes, for power management purposes. Each mode is characterized by a different amount of power consumption. For example, the StrongARM consumes 50mW of power in the *Idle* mode, and just 0.16mW in the *Sleep* mode. However, transitioning between operating modes involves a power and latency overhead. Thus, the power consumption levels of the various modes, the transition costs, and the amount of time spent by the MCU in each mode, all have a significant bearing on the total energy consumption (battery lifetime) of the sensor node.

RADIO

The sensor node's radio enables wireless communication with neighboring nodes and the outside world. There are several factors that affect the power consumption characteristics of a radio, including the type of modulation scheme used, data rate, transmit power (determined by the transmission distance), and the operational duty cycle. In general, radios can operate in four distinct modes of operation, namely *Transmit*, *Receive*, *Idle*, and *Sleep* modes. An important observation in the case of most radios is that, operating in *Idle* mode results in significantly high power consumption, almost equal to the power consumed in the *Receive* mode [11]. Thus, it is important to completely shutdown the radio rather than transitioning to *Idle* mode, when it is not transmitting or receiving data. Another influencing factor is that, as the radio's operating mode changes, the transient activity in the radio electronics causes a significant amount of power dissipation. For example, when the radio switches from sleep mode to transmit mode to send a packet, a significant amount of power is consumed for starting up the transmitter itself [9].

SENSORS

Sensor transducers translate physical phenomena to electrical signals, and can be classified as either analog or digital devices depending on the type of output they produce. There exist a diversity of sensors that measure environmental parameters such as temperature, light intensity, sound, magnetic fields, image *etc*. There are several sources of power consumption in a sensor, including (i) signal sampling and conversion of physical signals to electrical ones, (ii) signal conditioning, and (iii) analog to digital conversion. Given the diversity of sensors there is no typical power consumption number. In general, however, passive sensors such as temperature, seismic *etc*., consume negligible power relative to other components of sensor node. However, active sensors such as sonar rangers, array sensors such as imagers, and narrow field-of-view sensors that require repositioning such as cameras with pan-zoom-tilt can be large consumers of power.

POWER ANALYSIS OF SENSOR NODES

Table I shows the power consumption characteristics of Rockwell's WINS node [10], which represents a high-end sensor node, and is equipped with a powerful StrongARM SA-1100 processor from Intel, a radio module from Conexant Systems, and several sensors including acoustic and seismic ones. Table II gives the characteristics of the MEDUSA-II, an experimental sensor node developed at the Networked and Embedded Systems Lab, UCLA. The MEDUSA node, designed to be ultra low power, is a low-end sensor node similar to the COTS Motes developed as part of the SmartDust project [2]. It is equipped with an AVR microcontroller from ATMEL, a low-end RFM radio module, and a few sensors. As can be seen from the tables, the power dissipation characteristics of the two nodes differ significantly. There are several inferences that can be drawn from these tables:

> Using low-power components and trading off unnecessary performance for power savings during node design, can have a significant impact, up to a few orders of magnitude.

The node power consumption is strongly dependent on the operating modes of the components. For example, as Table I shows, the WINS node consumes only around one sixth the power when the MCU is in *Sleep* mode, than when it is in *Active* mode.

Due to extremely small transmission distances, the power consumed while receiving data can often be greater than the power consumed while transmitting packets, as is evident from Figure 2. Thus, conventional network protocols which usually assume the receive power to be negligible, are no longer efficient for sensor networks, and customized protocols which explicitly account for receive power have to be developed instead.

The power consumed by the node with the radio in *Idle* mode is approximately the same with the radio in *Receive* mode. Thus, operating the radio in *Idle* mode does not provide any advantage in terms of power. Previously proposed network protocols have often ignored this fact, leading to fallacious savings in power consumption, as pointed out in [11]. Therefore, the radio should be completely shut off whenever possible, to obtain energy savings.

BATTERY ISSUES

The battery supplies power to the complete sensor node, and hence plays a vital role in determining sensor node lifetime. Batteries are complex devices whose operation depends on many factors including battery dimensions, type of electrode material used, and diffusion rate of the active materials in the electrolyte. In addition, there can be several non-idealities that can creep in during battery operation, which adversely affect system lifetime. We describe the various battery non-idealities, and discuss system level design approaches that can be used to prolong battery lifetime.

Rated capacity effect

The most important factor that affects battery lifetime is the discharge rate or the amount of current drawn from the battery. Every battery has a rated current capacity, specified by the manufacturer. Drawing higher current than the rated value leads to a significant reduction in battery life. This is because, if a high current is drawn from the battery, the rate at which active ingredients diffuse through the electrolyte falls behind the rate at which they are consumed at the electrodes. If the high discharge rate is maintained for a long time, the electrodes run out of active materials, resulting in battery death even though active ingredients are still present in the electrolyte. Hence, to avoid battery life degradation, the amount of current drawn from the battery should be kept under tight check. Unfortunately, depending on the battery type (Lithium Ion, NiMH, NiCd, Alkaline, *etc.*), the minimum required current consumption of sensor nodes often exceeds the rated current capacity, leading to sub-optimal battery lifetime.

Relaxation effect

The effect of high discharge rates can be mitigated to a certain extent through battery relaxation. If the discharge current from the battery is cut off or reduced, the diffusion and transport rate of active materials catches up with the depletion caused by the discharge. This phenomenon is called the relaxation effect, and enables the battery to recover a portion of its lost capacity. Battery lifetime can be significantly increased if the system is operated such that the current drawn from the battery is frequently reduced to very low values, or is completely shut off [12].

DC-DC CONVERTER

The DC-DC converter is responsible for providing a constant supply voltage to the rest of the sensor node while utilizing the complete capacity of the battery. The efficiency factor associated with the converter plays a big role in determining battery lifetime [13]. A low efficiency factor leads to significant energy loss in the converter, reducing the amount of energy available to other sensor node components. Also, the voltage level across the battery terminals constantly decreases as it gets discharged. The converter therefore draws increasing amounts of current from the battery to maintain a constant supply voltage to the sensor node. As a result, the current drawn from the battery becomes progressively higher than the current that actually gets supplied to the rest of the sensor node. This leads to depletion in battery life due to the rated capacity effect, as explained earlier. Figure 3 shows the difference in current drawn from the battery and the current delivered to the sensor node for a Lithium-Ion coin cell battery.

NODE LEVEL ENERGY OPTIMIZATION

Having studied the power dissipation characteristics of wireless sensor nodes, we now focus our attention to the issue of minimizing the power consumed by these nodes. As a first step towards incorporating energy awareness into the network, it is necessary to develop hardware/software design methodologies and system architectures that enable energy-aware design and operation of individual sensor nodes in the network.

POWER-AWARE COMPUTING

Advances in low-power circuit and system design [6] have resulted in the development of several ultra low

power microprocessors, and microcontrollers. In addition to using low-power hardware components during sensor node design, operating the various system resources in a power-aware manner through the use of dynamic power management (DPM) [14] can reduce energy consumption further, increasing battery lifetime. A commonly used power management scheme is based on idle component shutdown, in which the sensor node, or parts of it, is shutdown or sent into one of several low-power states if no interesting events occur. Such event-driven power management is extremely crucial in maximizing node lifetime. The core issue in shutdown based DPM is deciding the state transition policy [14], since different states are characterized by different amounts of power consumption, and state transitions have a non-negligible power and time overhead.

While shutdown techniques save energy by turning off idle components, additional energy savings are possible in active state through the use of dynamic voltage scaling (DVS) [15]. Most microprocessor-based systems have a time varying computational load, and hence peak system performance is not always required. DVS exploits this fact by dynamically adapting the processor's supply voltage and operating frequency to just meet the instantaneous processing requirement, thus trading off unutilized performance for energy savings. DVS based power management, when applicable, has been shown to have significantly higher energy efficiency compared to shutdown based power management due to the convex nature of the energy- speed curve [15]. Several modern processors such as Intel's StrongARM and Transmeta's Crusoe support scaling of voltage and frequency, thus providing control knobs for energy-performance management.

For example, consider the target-tracking application discussed earlier. The duration of node shutdown can be used as a control knob to trade off tracking fidelity against energy. A low operational duty cycle for a node reduces energy consumption at the cost of a few missed detections. Further, the target update rate varies, depending on the Quality of Service requirements of the user. A low update rate implies more available latency to process each sensor data sample, which can be exploited to reduce energy through the use of DVS.

ENERGY AWARE SOFTWARE

Despite the higher energy efficiency of application specific hardware platforms, the advantage of flexibility offered by microprocessor and DSP based systems has resulted in the increasing use of programmable solutions during system design. Sensor network lifetime can be significantly enhanced if the system software, including the operating system (OS), application layer, and network protocols are all designed to be energy aware.

The OS is ideally poised to implement shutdown-based and DVS-based power management policies, since it has global knowledge of the performance and fidelity requirements of all the applications, and can directly control the underlying hardware resources, fine tuning the available performance-energy control knobs. At the core of the OS is a task scheduler, which is responsible for scheduling a given set of tasks to run on the system while ensuring that timing constraints are satisfied. System lifetime can be increased considerably by incorporating energy awareness into the task scheduling process [16], [17].

The energy aware real-time scheduling algorithm proposed in [16] exploits two observations about the operating scenario of wireless systems, to provide an adaptive power *vs.* fidelity tradeoff. The first observation is that these systems are inherently designed to operate resiliently in the presence of varying fidelity in the form of data losses, and errors over wireless links. This ability to adapt to changing fidelity is used to trade off against energy. Second, these systems exhibit significant correlated variations in computation and communication processing load due to underlying time-varying physical phenomena. This observation is exploited to proactively manage energy resources by predicting processing requirements. The voltage is set according to predicted computation requirements of individual task instances, and adaptive feedback control is used to keep the system fidelity (timing violations) within specifications.

The energy-fidelity tradeoff can be exploited further by designing the application layer to be energy scalable. This can be achieved by transforming application software such that the most significant computations are performed first. Thus, terminating the algorithm prematurely due to energy constraints, does not impact the result severely. For example, the target tracking application described earlier involves the extensive use of signal filtering algorithms such as Kalman filtering. Transforming the filtering algorithms to be energy scalable, trades off computational precision (and hence, tracking precision) for energy consumption. Several transforms to enhance the energy scalability of DSP algorithms are presented in [18].

POWER MANAGEMENT OF RADIOS

While power management of embedded processors has been studied extensively, incorporating power awareness into radio subsystems has remained relatively unexplored. Power management of radios is extremely important since wireless communication is a major power consumer during system operation. One way of characterizing the importance of this problem is in terms of the ratio of the energy spent in sending one bit to the energy spent in executing one instruction. While it is not

quite fair to compare this ratio across nodes without normalizing for transmission range, bit error probability, and the complexity of instruction (8-bit vs. 32-bit), this ratio is nevertheless useful. Example values are from 1500 to 2700 for Rockwell's WIN nodes, 220 to 2900 for the MEDUSA II nodes, and is around 1400 for the WINS NG 2.0 nodes from the Sensoria Corporation that are used by many researchers.

The power consumed by a radio has two main components to it, an RF component that depends on the transmission distance and modulation parameters, and an electronics component that accounts for the power consumed by the circuitry that performs frequency synthesis, filtering, up-converting, *etc*. Radio power management is a non-trivial problem, particularly since the well-understood techniques of processor power management may not be directly applicable. For example, techniques such as dynamic voltage and frequency scaling reduce processor energy consumption at the cost of an increase in the latency of computation. However, in the case of radios, the electronics power can be comparable to the RF component (which varies with the transmission distance). Therefore, slowing down the radio may actually lead to an increase in energy consumption. Other architecture specific overheads like the startup cost of the radio can be quite significant [9], making power management of radios a complex problem. The various tradeoffs involved in incorporating energy awareness into wireless communication will be discussed further in the next section.

ENERGY AWARE PACKET FORWARDING

In addition to sensing and communicating its own data to other nodes, a sensor node also acts as a router, forwarding packets meant for other nodes. In fact, for typical sensor network scenarios, a large portion (around 65%) of all packets received by a sensor node need to be forwarded to other destinations [19]. Typical sensor node architectures implement most of the protocol processing functionality on the main computing engine. Hence, every received packet, irrespective of its final destination, travels all the way to the computing subsystem and gets processed, resulting in a high energy overhead. The use of intelligent radio hardware, as shown in Figure 4, enables packets that need to be forwarded to be identified and re-directed from the communication subsystem itself, allowing the computing subsystem to remain in *Sleep* mode, saving energy [19].

ENERGY AWARE WIRELESS COMMUNICATION

While power management of individual sensor nodes reduces energy consumption, it is important for the communication between nodes to be conducted in an energy efficient manner as well. Since the wireless transmission of data accounts for a major portion of the total energy consumption, power management decisions that take into account the effect of inter-node communication yield significantly higher energy savings. Further, incorporating power management into the communication process enables the diffusion of energy awareness from an individual sensor node to a group of communicating nodes, thereby enhancing the lifetime of entire regions of the network. To achieve power-aware communication it is necessary to identify and exploit the various performance-energy trade-off knobs that exist in the communication subsystem.

MODULATION SCHEMES

Besides the hardware architecture itself, the specific radio technology used in the wireless link between sensor nodes plays an important role in energy considerations. The choice of modulation scheme greatly influences the overall energy versus fidelity and latency tradeoff that is inherent to a wireless communication link. Equation (1) expresses the energy cost for transmitting one bit of information, as a function of the packet payload size L, the header size H, the fixed overhead E_{start} associated with the radio startup transient, and the symbol rate R_S for an M-ary modulation scheme [9], [20]. P_{elec} represents the power consumption of the electronic circuitry for frequency synthesis, filtering, modulating, upconverting, *etc*. The power delivered by the power amplifier, P_{RF}, needs to go up as M increases, in order to maintain the same error rate.

$$Ebit = \frac{E_{start}}{L} + \frac{P_{elec} + P_{RF}(M)}{R_S * \log_2 M} * (1 + \frac{H}{L}) \quad (1)$$

Figure 5 plots the communication energy per bit as a function of the packet size and the modulation level M. This curve was obtained using the parameters given in Table III, which are representative for sensor networks, and choosing Quadrature Amplitude Modulation (QAM) [9], [20]. The markers in Figure 5 indicate the optimal modulation setting for each packet size, which is independent of L. In fact, this optimal modulation level is relatively high, close to 16-QAM for the values specified in Table III. Higher modulation levels might be unrealistic in low-end wireless systems, such as sensor nodes. In these scenarios, a practical guideline for saving energy is to transmit as fast as possible, at the optimal setting [9]. However, if for reasons of peak-throughput, higher modulation levels than the optimal one need to be provided, adaptively changing the modulation level can lower the overall energy consumption. When the

instantaneous traffic load is lower than the peak value, transmissions can be slowed down, possibly all the way to the optimal operating point. This technique of dynamically adapting the modulation level to match the instantaneous traffic load, as part of the radio power management, is called modulation scaling [20]. It is worth noting that dynamic modulation scaling is the exact counterpart of dynamic voltage scaling, which has been shown to be extremely effective for processor power management, as described earlier.

The above conclusions are expected to hold for other implementations of sensor network transceivers as well. Furthermore, since the startup cost is significant in most radio architectures [9], it is beneficial to operate with as large a packet size as possible, since it amortizes this fixed overhead over more bits. However, aggregating more data into a single packet has the downside of increasing the overall latency of information exchange.

The discussion up until now has focused on the links between two sensor nodes, which are characterized by their short distance. However, when external users interact with the network, they often times do so via specialized gateway nodes [22], [23]. These gateway nodes offer long-haul communication services, and are therefore in a different regime where P_{RF} dominates P_{elec}. In this case, the optimal M shifts to the lowest possible value, such that it becomes beneficial to transmit as slow as possible, subject to the traffic load. In this regime, modulation scaling is clearly very effective [20].

COORDINATED POWER MANAGEMENT TO EXPLOIT COMPUTATION COMMUNICATION TRADEOFF

Sensor networks involve several node-level and network-wide computation-communication tradeoffs, which can be exploited for energy management. At the individual node level, power management techniques such as DVS and modulation scaling reduce the energy consumption at the cost of increased latency. Since both the computation and communication subsystems take from the total acceptable latency budget, exploiting the inherent synergy between them to perform coordinated power management will result in far lower energy consumption. For example, the relative split up of the available latency for the purposes of dynamic voltage scaling and dynamic modulation scaling significantly impacts the energy savings obtained. Figure 6 shows a system power management module that is integrated into the OS, and performs coordinated power management of the computing, communication and sensing subsystems.

The computation-communication tradeoff manifests itself in a powerful way due to the distributed nature of these sensor networks. The network's inherent capability for parallel processing offers further energy optimization potential. Distributing an algorithm's computation among multiple sensor nodes enables the computation to be performed in parallel. The increased allowable latency per computation enables the use of voltage scaling, or other energy-latency tradeoff techniques. Distributed computing algorithms however demand more inter-node collaboration, thereby increasing the amount of communication that needs to take place.

These computation-communication tradeoffs extend beyond individual nodes to the network level too. As we will discuss in the next section, the high redundancy present in the data gathering process, enables the use of data combining techniques to reduce the amount of data to be communicated, at the expense of extra computation at individual nodes to perform data aggregation.

LINK LAYER OPTIMIZATIONS

While exploring energy-performance-quality tradeoffs, reliability constraints also have to be considered, which are related to the interplay of communication packet losses and sensor data compression. Reliability decisions are usually taken at the link layer, which is responsible for some form of error detection and correction. Adaptive error correction schemes were proposed in [24] to reduce energy consumption, while maintaining the bit error rate (BER) specifications of the user. For a given BER requirement, error control schemes reduce the transmit power required to send a packet, at the cost of additional processing power at the transmitter and receiver. This is especially useful for long-distance transmissions to gateway nodes, which involve large transmit power. Link layer techniques also play an indirect role in reducing energy consumption. The use of a good error control scheme minimizes the number of times a packet retransmissions, thus reducing the power consumed at the transmitter as well as the receiver.

NETWORK-WIDE ENERGY OPTIMIZATION

Incorporating energy awareness into individual nodes and pairs of communicating nodes alone does not solve the energy problem in sensor networks. The network as a whole should be energy-aware, for which the network-level global decisions should be energy-aware.

TRAFFIC DISTRIBUTION

At the highest level of sensor network, the issue of how traffic is forwarded from the data source to the data sink arises. Data sinks typically are user nodes or specialized gateways that connect the sensor network to the outside world. One aspect of traffic forwarding is the choice of an energy efficient multi-hop route between source and destination. Several approaches have been

proposed [3], [23], [25] which aim at selecting a path that minimizes the total energy consumption.

However, such a strategy does not always maximize the network lifetime [26]. Consider the target-tracking example again. While forwarding the gathered and processed data to the gateway, it is desirable to avoid routes through regions of the network that are running low on energy resources, thus preserving them for future, possibly critical detection and communication tasks. For the same reason, it is in general, undesirable to continuously forward traffic via the same path, even though it minimizes the energy, up to the point where the nodes on that path are depleted of energy, and the network connectivity is compromised. It would, instead, be preferable to spread the load more uniformly over the network. This general guideline can increase the network lifetime in typical scenarios, although this is not always the case [26] as the optimal distribution of traffic load is possible only when future network activity is known.

TOPOLOGY MANAGEMENT

The traffic distribution through appropriate routing essentially exploits the macro-scale redundancy of possible routes between source and destination. However, on each route, there is also a micro-scale redundancy of nodes that are essentially equivalent for the multi-hop path. In typical deployment scenarios, a dense network is required to ensure adequate coverage of both the sensing and multi-hop routing functionality, in addition to improving network fault-tolerance [11], [27]. It is immediately apparent that there exist several adaptive energy-fidelity tradeoffs here too. For example, in target tracking, denser distributions of sensors lead to increasingly precise tracking results. However, if network lifetime is more critical than tracking precision, tracking could be done using data samples from fewer nodes. In addition to reducing the computational complexity itself, this also reduces the communication requirements of the non- participating nodes since they no longer have to send in their data to be processed.

Despite the inherent node redundancy, these high densities do not immediately result in an increased network lifetime, as the radio energy consumption in *Idle* mode does not differ much from that in *Transmit* or *Receive* mode. Only by transitioning the radio to the *Sleep* state can temporarily quiescent nodes conserve battery energy. However, in this state, nodes cannot be communicated with, and have effectively retracted from the network, thereby changing the active topology. Thus, the crucial issue is to intelligently manage the sleep state transitions while providing robust undisturbed operation.

This reasoning is the foundation for the time slotted MAC protocol for sensor networks in [22] where the nodes only need to wake up during time slots that they are assigned to, although this comes at the cost of maintaining time synchronization. An alternative approach advocates explicit node wake up via a separate, but low-power paging channel. In addition, true topology management explicitly leverages the fact that in high node density several nodes can be considered backups of each other with respect to traffic forwarding. The GAF protocol [11] identifies equivalent nodes based on their geographic location in a virtual grid such that they replace each other directly and transparently in the routing topology. In SPAN [27], a limited number of coordinator nodes are elected to forward the bulk of the traffic as a backbone within the ad-hoc network, while other nodes can frequently transition to a sleep state. Both GAF and SPAN are distributed protocols that provision for periodic rotation of node functionality to ensure fair energy consumption distribution. STEM [28] goes beyond GAF and SPAN in improving the network lifespan by exploiting the fact that most of the time the network is only sensing its environment waiting for an event to happen. By eliminating GAF and SPAN's restriction of network capacity preservation at all times, STEM trades off an increased latency to set up a multi-hop path to achieve much higher energy savings.

COMPUTATION COMMUNICATION TRADEOFFS

Intelligent routing protocols and topology management ensure that the burden of forwarding traffic is distributed between nodes in an energy-efficient, *i.e.*, network lifetime improving, fashion. Further enhancements are possible by reducing the size of the packets that are forwarded. As mentioned earlier, each node already processes its sensor data internally to this end. Consider the target tracking application. Due to high node densities, a target is detected not only by a single node, but also by an entire cloud of nearby nodes, leading to a high degree of redundancy in the gathered data. Combining the information from the nodes in this cloud via in-network processing can both improve the reliability of the detection event/data, and greatly reduce the amount of traffic. One option is to combine the sensor readings of different nodes in a coherent fashion via beam-forming techniques [22]. Alternatively, non-coherent combining, also known as data fusion or aggregation, can be used, which does not require synchronization, but is less powerful. Several alternatives have been proposed to select the nodes that perform the actual combining, such as winner election [22], clustering [23], or traffic-steered [26]. These techniques illustrate the effectiveness of exploiting network wide computation-communication tradeoffs.

OVERHEAD REDUCTION

The sensor data packet payload can be quite compact due to in-network processing, with reported packet payloads as low as 8 to 16 bits [22]. Also, attribute based naming and routing are being used [3], where the more common attributes can be coded in fewer bits. Short random identifiers have been proposed to replace unique identifiers for end-to-end functions such as fragmentation/reassembly. Spatial reuse, combined with Huffman-coded representation, can significantly reduce the size of MAC addresses compared to traditional network-wide unique identifiers [21]. Packet headers using attribute-based routing identifiers and encoded reusable MAC addresses are very compact, of the order of 10 bits. This reduction will become more important as radios with smaller startup cost are developed [9].

CONCLUSIONS

Sensor networks have emerged as a revolutionary technology for querying the physical world and hold promise in a wide variety of applications. However, the extremely energy constrained nature of these networks necessitate that their design and operation be done in an energy-aware manner, enabling the system to dynamically make tradeoffs between performance, fidelity, and energy consumption. We presented several energy optimization and management techniques at node, link, and network level, leveraging which can lead to significant enhancement in sensor network lifetime.

ACKNOWLEDGEMENTS

This paper is based in part on research funded through DARPA's PAC/C and SensIT programs under AFRL contracts F30602-00-C-0154 and F30602-99-1-0529 respectively, and through NSF Grants ANI-0085773 and MIPS-9733331. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the author(s), and do not necessarily reflect the views of DARPA, AFRL, or NSF. The authors would like to acknowledge their colleagues at the Networked and Embedded Systems Laboratory, UCLA for several interesting and stimulating technical discussions.

REFERENCES

[1] Wireless Integrated Network Sensors, University of California, Los Angeles. (http://www.janet.ucla.edu/WINS)

[2] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: mobile networking for smart dust", in Proc. Mobicom, pp. 483-492, 1999.

[3] D. Estrin and R. Govindan, "Next century challenges: scalable coordination in sensor networks", in *Proc. Mobicom*, pp. 263-270, 1999.

[4] A. P. Chandrakasan, et al., "Design considerations for distributed microsensor systems", in *Proc. CICC*, 1999, pp. 279-286.

[5] J. Rabaey, et al., "PicoRadio supports ad hoc ultra low power wireless networking", in *IEEE Computer*, July 2000, pp. 42-48.

[6] A. P. Chandrakasan and R. W. Brodersen, Low Power CMOS Digital Design, Kluwer Academic Publishers, Norwell, MA, 1996.

[7] V. Tiwari, S. Malik, A. Wolfe, and M. T. C. Lee, "Instruction level power analysis and optimization of software", in *Jrnl. VLSI Signal Processing*, Kluwer Academic Publishers, pp. 1-18, 1996.

[8] A. Sinha and A. P. Chandrakasan, "Jouletrack: A web based tool for software energy profiling", in *Proc. Design Automation Conf.*, 2001.

[9] A. Wang, S-H. Cho, C. G. Sodini, and A. P. Chandrakasan, "Energy-efficient modulation and MAC for asymmetric microsensor systems", in *Proc. ISLPED*, 2001.

[10] WINS project, Rockwell Science Center, (http://wins..rsc.rockwell.com).

[11] Y. Xu, J. Heidemann and D. Estrin, "Geography-informed energy conservation for ad hoc routing", in *Proc. Mobicom*, 2001.

[12] C. F. Chiasserini and R. R. Rao, "Pulsed battery discharge in communication devices", in *Proc. Mobicom*, 1999.

[13] S. Park, A. Savvides, and M. Srivastava, "Battery capacity measurement and analysis using lithium coin cell battery", in *Proc. ISLPED*, 2001.

[14] L. Benini and G. DeMicheli, Dynamic Power Management: Design Techniques & CAD Tools, Kluwer Academic Publishers, Norwell, MA, 1997.

[15] T. A. Pering, T. D. Burd, and R. W. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms", in *Proc. ISLPED*, pp. 76-81, 1998.

[16] V. Raghunathan, P. Spanos, and M. Srivastava, "Adaptive power-fidelity in energy aware wireless embedded systems", to be presented at *IEEE Real Time Systems Symposium*, 2001.

[17] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy", in *Proc. Annual Symp. on Foundations of Computer Science*, pp.374-382, 1995.

[18] A. Sinha, A. Wang, and A. P. Chandrakasan, "Algorithmic transforms for efficient energy scalable computation", in *Proc. ISLPED*, 2000.

[19] V. Tsiatsis, S. Zimbeck, and M. Srivastava, "Architectural strategies for energy efficient packet forwarding in wireless sensor networks", in *Proc. ISLPED*, 2001.

[20] C. Schurgers, O. Aberthorne, and M. Srivastava, "Modulation scaling for energy aware communication systems", in *Proc. ISLPED*, 2001.

[21] C. Schurgers, G. Kulkarni, and M. Srivastava, "Distributed assignment of encoded MAC addresses in wireless sensor networks", in *Proc.MobiHoc*, 2001.

[22] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for self-organization of a wireless sensor network", in *IEEE Personal Comm. Magazine*, vol.7, no.5, pp. 16-27, Oct. 2000.

[23] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks", in *Proc. Hawaii Intl. Conf. on System Sciences*, Hawaii, 2000.

[24] P. Lettieri, C. Fragouli, and M. Srivastava, "Low power error control for wireless links", in *Proc. Mobicom*, pp. 139-150, 1997.

[25] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks", in *Proc. INFOCOM*, 2000.

[26] C. Schurgers and M. Srivastava, "Energy efficient routing in sensor networks", in *Proc. Milcom*, 2001.

[27] B. Chen, K. Jamieson, H. Balakrishnan, and R Morris, "SPAN: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, in *Proc. Mobicom*, 2001.

[28] C. Schurgers, V. Tsiatsis, and M. Srivastava, "STEM: Topology management for energy efficient sensor networks," To appear in the Proceedings of the 2002 IEEE Aerospace Conference, March 2002.

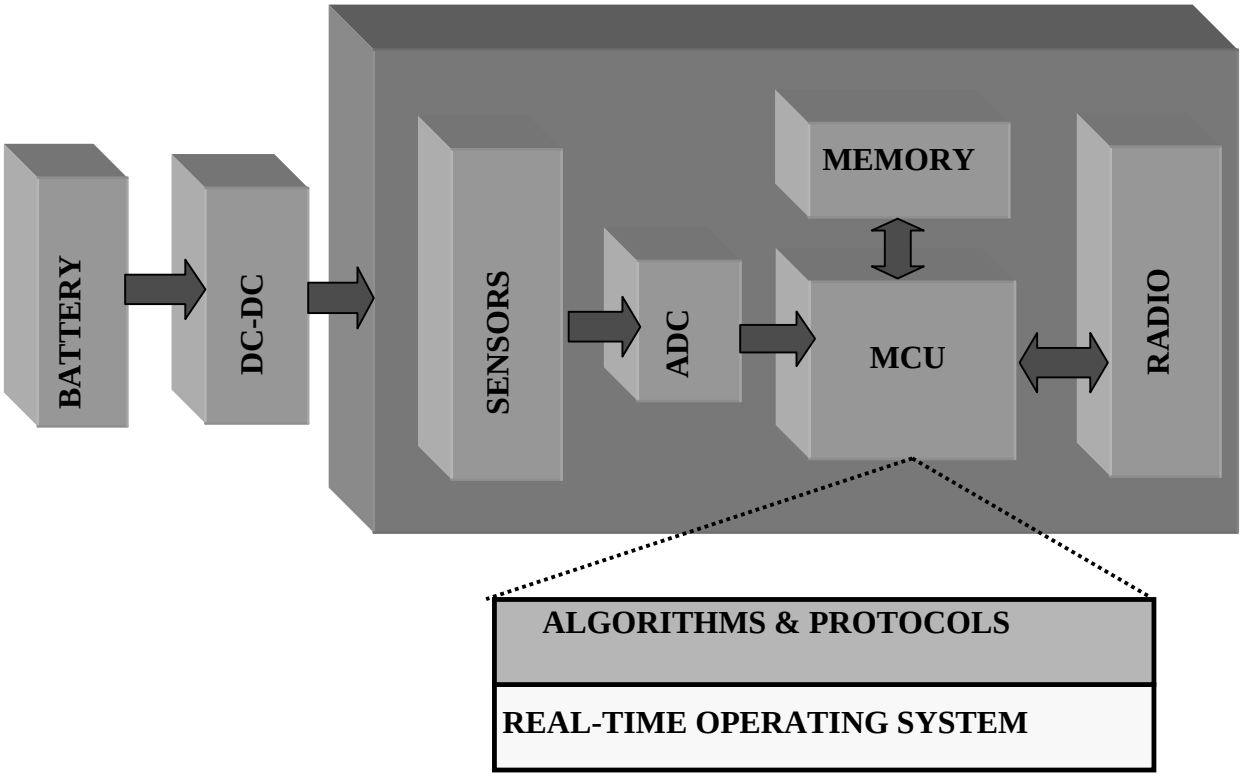


Fig 1. System architecture of a typical wireless sensor node

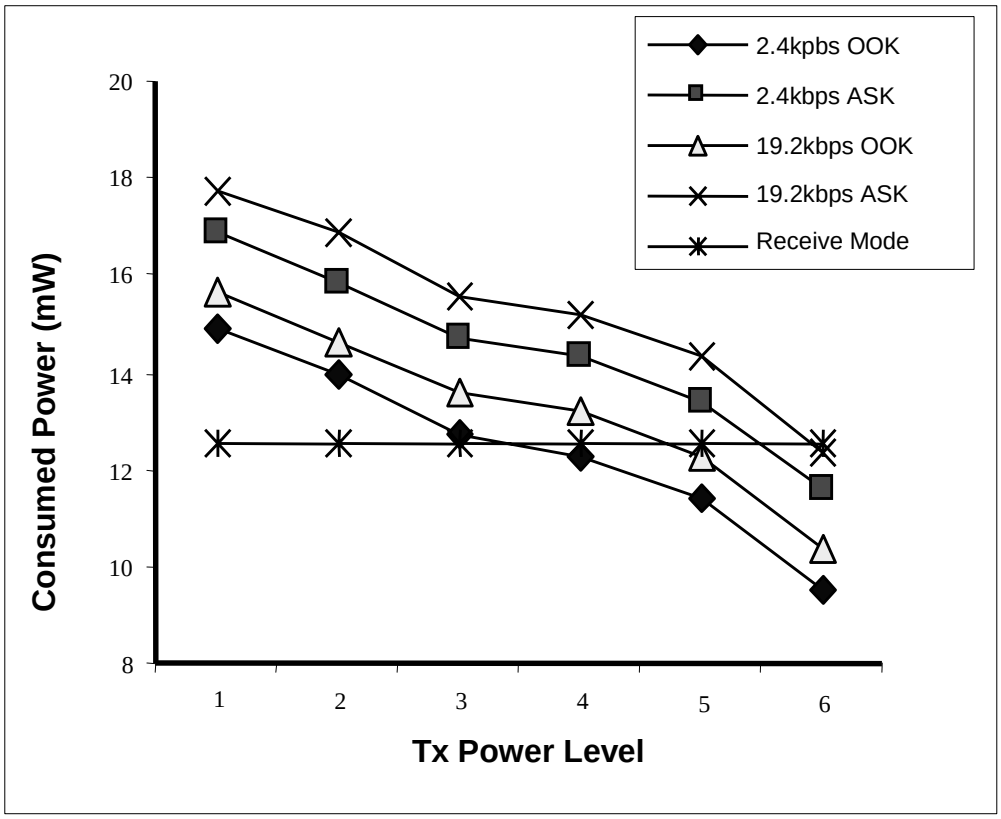


Fig. 2. Power consumption of an RFM radio in various modes of operation

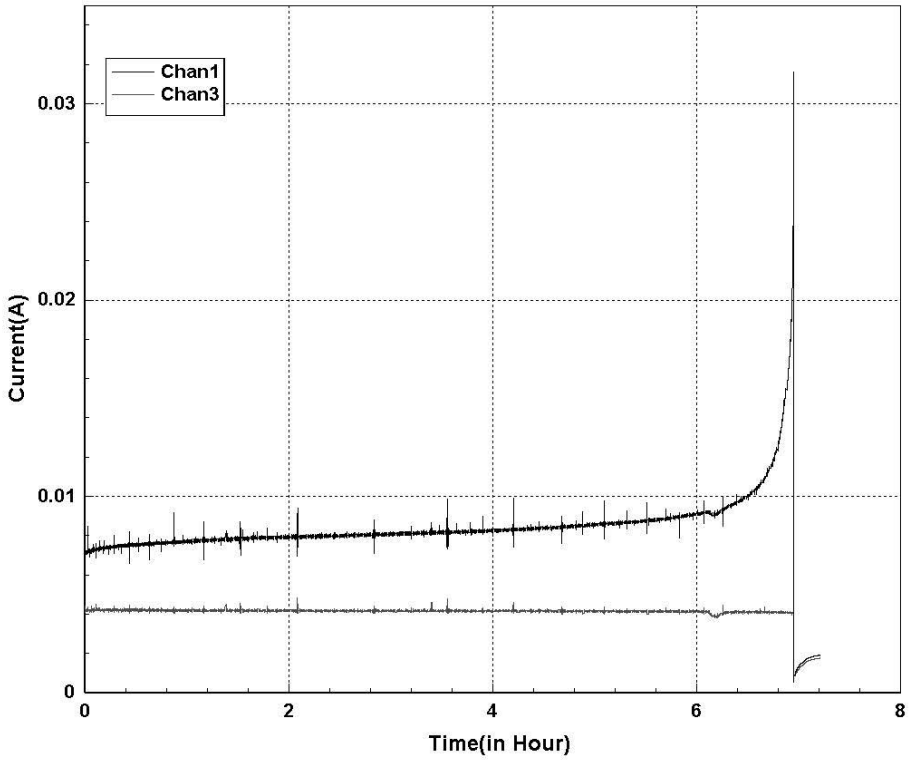


Fig. 3. Current drawn from the battery (Chan 1) and current supplied to the node (Chan 3)

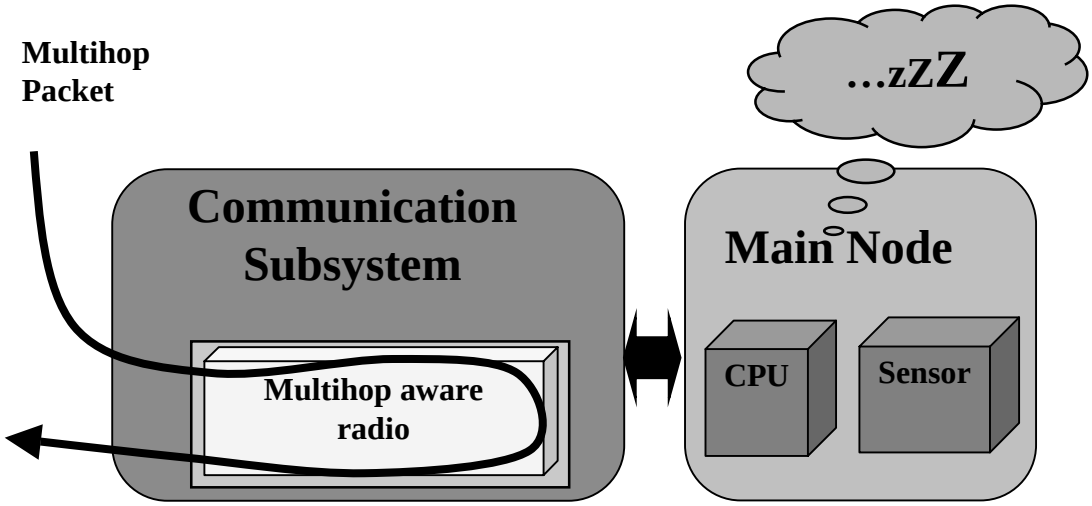


Fig. 4. Energy-aware packet forwarding architecture

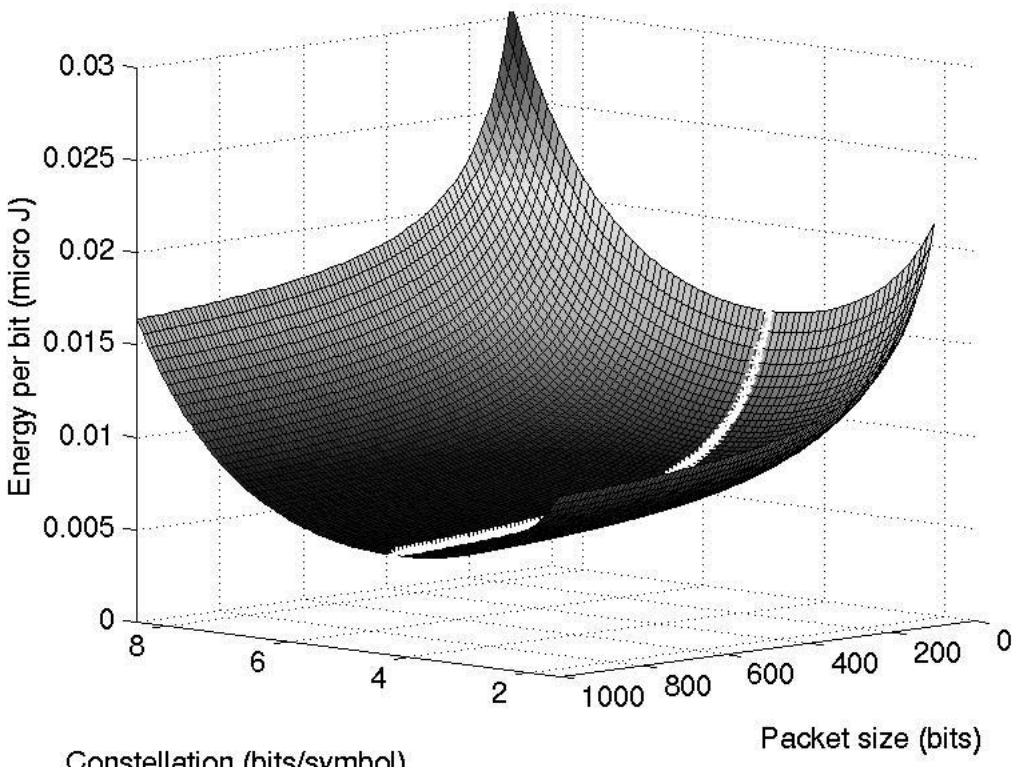


Fig. 5. Radio energy per bit as a function of packet size and modulation level

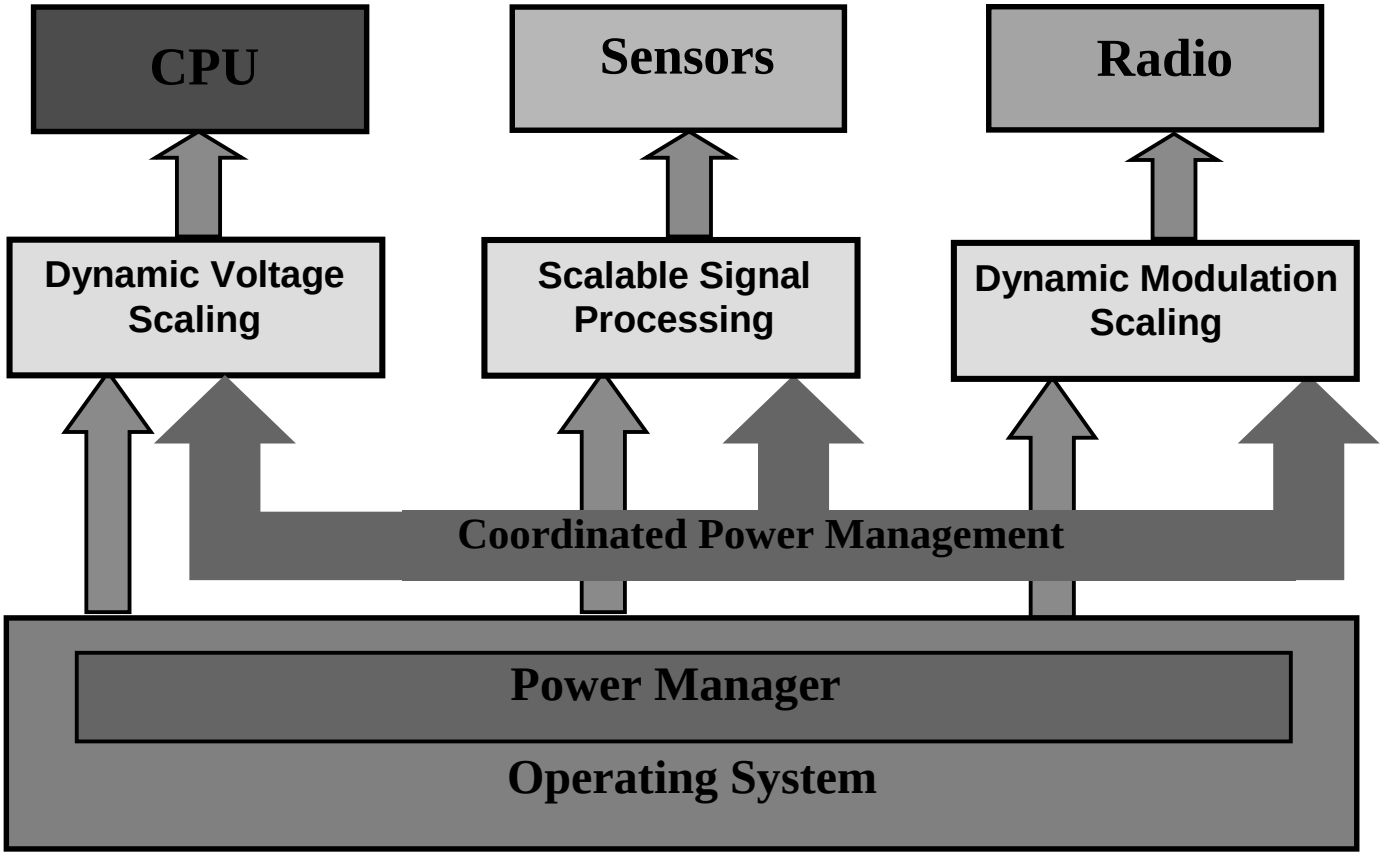


Fig. 6. Coordinated power management at the node level to exploit computation-communication tradeoffs

TABLE I

POWER ANALYSIS OF ROCKWELL'S WINS NODES

MCU Mode	**Sensor Mode**	**Radio Mode**	**Power (mW)**
Active	On	Tx (Power: 36.3 mW)	1080.5
		Tx (Power: 19.1 mW)	986.0
		Tx (Power: 13.8 mW)	942.6
		Tx (Power: 3.47 mW)	815.5
		Tx (Power: 2.51 mW)	807.5
		Tx (Power: 0.96 mW)	787.5
		Tx (Power: 0.30 mW)	773.9
		Tx (Power: 0.12 mW)	771.1
Active	On	Rx	751.6
Active	On	Idle	727.5
Active	On	Sleep	416.3
Active	On	Removed	383.3
Sleep	On	Removed	64.0
Active	Removed	Removed	360.0

TABLE II

POWER ANALYSIS OF MEDUSA II NODES

MCU Mode	Sensor Mode	Radio Mode	Mod. Scheme	Data Rate	Power (mW)
Active	On	Tx (Power: 0.7368 mW)	OOK	2.4 kbps	24.58
		Tx (Power: 0.0979 mW)	OOK	2.4 kbps	19.24
		Tx (Power: 0.7368 mW)	OOK	19.2 kbps	25.37
		Tx (Power: 0.0979 mW)	OOK	19.2 kbps	20.05
		Tx (Power: 0.7368 mW)	ASK	2.4 kbps	26.55
		Tx (Power: 0.0979 mW)	ASK	2.4 kbps	21.26
		Tx (Power: 0.7368 mW)	ASK	19.2 kbps	27.46
		Tx (Power: 0.0979 mW)	ASK	19.2 kbps	22.06
Active	On	Rx	Any	Any	22.20
Active	On	Idle	Any	Any	22.06
Active	On	Off	Any	Any	9.72
Idle	On	Off	Any	Any	5.92
Sleep	Off	Off	Any	Any	0.02

TABLE III

TYPICAL RADIO PARAMETERS FOR SENSOR NETWORKS

E_{start}	1∝J
P_{elec}	12mW
P_{RF}	1mW for 4-QAM
R_S	1 Mbaud
H	16 bits

Appendix H:

STEM: Topology Management for Energy Efficient Sensor Networks [2]

Curt Schurgers
University of California, Los Angeles
Eng. IV Bldg., UCLA-EE
Los Angeles, CA 90095
1-310-206-4465
curts@ee.ucla.edu

Vlasios Tsiatsis
University of California, Los Angeles
Eng. IV Bldg., UCLA-EE
Los Angeles, CA 90095
1-310-825-7707
tsiatsis@ee.ucla.edu

Mani B. Srivastava
University of California, Los Angeles
Eng. IV Bldg., UCLA-EE
Los Angeles, CA 90095
1-310-267-2098
mbs@ee.ucla.edu

Abstract—In wireless sensor networks, where energy efficiency is the key design challenge, the energy consumption is typically dominated by the node's communication subsystem. It can only be reduced significantly by transitioning the embedded radio to a sleep state, at which point the node essentially retracts from the network topology. Existing topology management schemes have focused on cleverly selecting which nodes can turn off their radio, without sacrificing the capacity of the network. We propose a new technique, called Sparse Topology and Energy Management (STEM), that dramatically improves the network lifetime by exploiting the fact that most of the time, the network is only sensing its environment waiting for an event to happen. By alleviating the restriction of network capacity preservation, we can trade off extensive energy savings for an increased latency to set up a multi-hop path. We will also show how STEM integrates efficiently with existing topology management techniques.

TABLE OF CONTENTS

1. INTRODUCTION

Sensor Networks

Sensor nodes are autonomous devices equipped with heavily integrated sensing, processing, and communication capabilities [1][2]. When these nodes are networked together in an ad-hoc fashion, they form a sensor network. The nodes gather data via their sensors, process it locally or coordinate amongst neighbors and forward the information to the user or, in general, a data sink. Due to the node's limited transmission range, this forwarding mostly involves using multi-hop paths through other nodes [3]. It is important to point out that a node in the network has essentially two different tasks: (1) sensing its environment and processing the information, and (2) forwarding traffic as an intermediate relay in the multi-hop path.

Such sensor networks find applicability in wildlife observation, smart office buildings, and applications such as battlefield or disaster area monitoring. They are also considered to establish sensor grids on distant planetary bodies, relaying information to the interplanetary Internet. In addition, future large-scale networks of resource limited satellites are likely to be governed by similar principles and can benefit from the design methodologies developed for sensor networks.

The major design challenge for this type of networks is to increase their operational lifetime as much as possible, despite the limited energy supply of each node [1][2][3]. Indeed, to provide unobtrusive operation, sensor nodes are miniature devices and, as a result, operate on a tiny, non-replaceable battery. Energy efficiency is therefore the critical design constraint.

In terms of energy consumption, the wireless exchange of data between nodes strongly dominates other node functions such as sensing and processing [1][3][4]. Moreover, the radio consumes almost as much energy in receive and idle mode as it does in transmit mode [4]. Significant energy savings are only obtainable by putting the node in sleep mode, essentially disconnecting it from the network and changing the topology. This has severe repercussions, as sleeping nodes can no longer function as relays in multi-hop paths.

Topology Management

The goal of topology management is to coordinate the sleep transitions of all the nodes, while ensuring adequate network connectivity, such that data can be forwarded efficiently to

0-7803-7231-X/01/$10.00/© 2002 IEEE
[2] IEEEAC paper #260, Updated Sept 24, 2001

the data sink. Existing topology management schemes try to do just that: they remove redundancy in the network topology while trying to conserve the data communication capacity [5][6]. Due to this restriction of sacrificing as little of the forwarding capacity as possible, the gains of these schemes are relatively modest, even for extremely dense networks. The underlying reasoning is that they implicitly assume the network has data to forward, which we refer to as being in the ***'transfer state'***.

However, most of the time, the sensor network is only monitoring its environment, waiting for an event to happen. For a large subset of sensor net applications, no data needs to be forwarded to the data sink in this ***'monitoring state'***.

Consider for example a sensor network that is designed to detect brush fires. It has to remain operational for months or years, while only sensing if a fire has started. Once a fire is detected, this information should be forwarded to the user quickly. Even when we want to track how the fire spreads, it probably suffices for the network to remain up only for an additional week or so. It is clear that although the transfer state should be energy efficient, it is far more important for the monitoring state to be ultra-low power, as the network resides in this state most of the time. Similar observations hold for applications such as surveillance of battlefields, machine failures, room occupancy, or other reactive scenarios, where the user needs to be informed once a condition is satisfied.

Of course, different parts of the network could be in monitoring or transfer state, so, strictly speaking, the 'state' is more a property of the locality of node, rather than the entire network. We also note that the network probably needs to transition to the transfer state periodically to exchange network management and maintenance messages [1].

Nevertheless, these sensor networks often spend the vast majority of time in the monitoring state. It is therefore critical to optimize the network's energy efficiency in this state as much as possible, beyond what is accomplished by existing topology management techniques.

We acknowledge that sensor networks could also be designed to periodically send updates to the data sink, or, in general, reside in the monitoring state much less frequent. In this case, the technique presented in this paper is expected to be much less useful. Yet, we foresee that the majority of sensor network applications and scenarios would have significant periods without data forwarding activity, and as such greatly benefit from the topology management technique we will present here.

Our Contributions

We observe that in the monitoring state, which we expect to be predominant, the requirement of capacity preservation is no longer pertinent. Instead, nodes only need the ability to wake up neighbors to perform coordinated sensing or set up a path, with a reasonable latency. As such, the network topology can be much sparser, and nodes spend more time sleeping.

In principle, the communication capacity could be reduced to virtually zero, by turning off the radios of all nodes (*i.e.*, putting them in the sleep mode). Note that the sensors and processor can be on at that time, since they are much less power hungry than the communication subsystem. As soon as events are detected, however, nodes need to be woken up quickly to set up the multi-hop communication path to the data sink. This requires nodes to communicate with each other, but this is only possible if they have their radio turned on. We obviously have two contradictory requirements here: on the one hand, nodes should be in sleep mode as often as possible when they are in the monitoring state, yet they should receive requests of other nodes to return to the more active transfer state.

In this paper, we propose a new topology management scheme, called ***STEM (Sparse Topology and Energy Management)***. It trades off energy consumption in the monitoring state, versus latency of switching back to the transfer state. The resulting energy savings have a significant impact on the network lifetime, which is extended in addition to and beyond existing approaches.

Prior Work

For sensor networks, two alternative routing approaches have been considered: flat multi-hop and clustering. Although STEM is applicable to both of them, we mainly focus on flat multi-hop routing [3][7][8]. For clustered approaches [9], which are possibly hierarchical, our scheme can be used to reduce the energy of the cluster heads, although the gains are expected to be less dramatic here.

Recently, topology management techniques, called SPAN [5] and GAF [6], have been proposed for flat multi-hop routing. They operate on the assumption that the network capacity needs to be preserved. As a result, the energy consumption is approximately the same whether the network is in the transfer or monitoring state, as no distinction is made between them. In contrast, STEM dramatically improves the energy efficiency in the monitoring state, far beyond what is achieved by SPAN and GAF alone, which can still be used in the transfer state. We can thus claim that STEM is in a way orthogonal to these existing techniques.

In SPAN [5], a limited set of nodes forms a multi-hop forwarding backbone, which tries to preserve the original capacity of the underlying ad-hoc network. Other nodes transition to sleep states more frequently, as they no longer carry the burden of forwarding data of other nodes. To balance out energy consumption, the backbone functionality is rotated between nodes, and as such there is a strong

interaction with the routing layer. Unlike SPAN, STEM does not try to conserve capacity, resulting in greater energy savings, and also does not impact routing.

Geographic Adaptive Fidelity (GAF) [6] exploits the fact that nearby nodes can perfectly and transparently replace each other in the routing topology. The sensor network is subdivided into small grids, such that nodes in the same grid are equivalent from a routing perspective. At each point in time, only one node in each grid is active, while the others are in the energy-saving sleep mode. Substantial energy gains are, however, only achieved in very dense networks. We will discuss this issue further on in this paper, when we integrate STEM with GAF.

An approach that is closely related to STEM is the use of a separate paging channel to wake up nodes that have turned of their main radio [10]. However, the paging channel radio cannot be put in the sleep mode for obvious reasons. This approach thus critically assumes that the paging radio is much lower power than the one used for regular data communications. It is yet unclear if such radio can be designed. STEM basically emulates the behavior of a paging channel, by having a radio with a low duty cycle radio, instead of a radio with low power consumption.

2. SPARSE TOPOLOGY MANAGEMENT

Basic Concept

In the application scenarios we consider in this paper, the sensor network is in the monitoring state the vast majority of its lifetime. Ideally, we would like to only turn on the sensors and some preprocessing circuitry. When a possible event is detected, the main processor is woken up to analyze the data in more detail. The radio, which is normally turned off, is only woken up if the processor decides that the information needs to be forwarded to the data sink.

Now, the problem is that the radio of the next hop in the path to the data sink is still turned off, if it did not detect that same event. As a solution, each node periodically turns on its radio for a short time to listen if someone wants to communicate with it. The node that wants to communicate, the ***'initiator node'***, sends out a beacon with the ID of the node it is trying to wake up, called the ***'target node'***. In fact, this can be viewed as the initiator node attempting to activate the link between itself and the target node. As soon as the target node receives this beacon, it responds to the initiator node and both keep their radio on at this point. If the packet needs to be relayed further, the target node will become the initiator node for the next hop and the process is repeated.

Dual Frequency Setup

Once both nodes that make up a link have their radio on, the link is active, and can be used for subsequent packets. In order for actual data transmissions not to interfere with the wakeup protocol, we propose to send them in different frequency bands using a separate radio in each band. Sensor nodes developed by Sensoria Corporation [11], for example, are already equipped with a dual radio.

Figure 1 shows the proposed radio setup. The wakeup messages, which were discussed in the subsection above, are transmitted by the radio operating in frequency band f_1. We refer to these communications as occurring in the ***'wakeup plane'***. Once the initiator node has successfully notified the target node, both nodes turn on their radio that operates in frequency band f_2. The actual data packets are transmitted in this band, or what we call the ***'data plane'***.

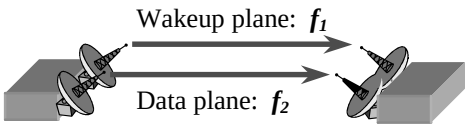


Figure 1 – Radio setup of a sensor node

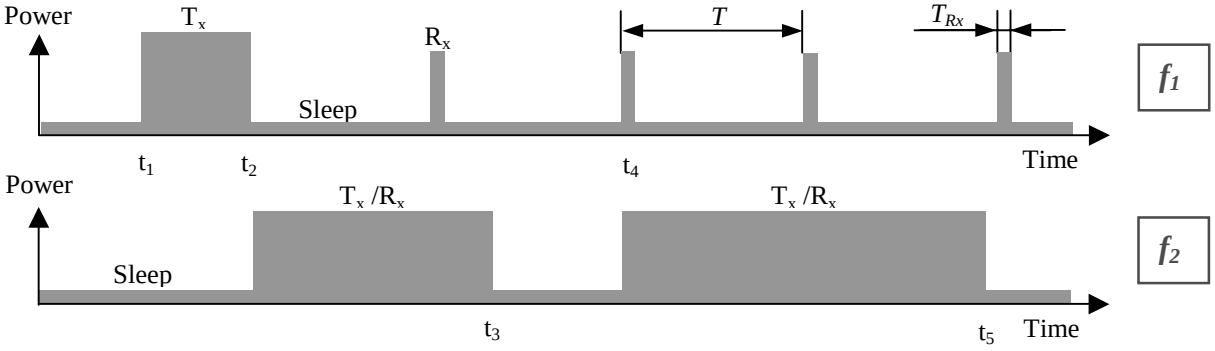


Figure 2 – State transitions of STEM for a particular node

STEM Operation

Figure 2 presents an example of typical radio mode transitions for one particular node in the network. Some representative power numbers for the different modes are summarized in Table 1. These numbers correspond to a 2.4 Kbps low-power RFM radio using OOK modulation, with an approximate transmit range of 20 meters [4].

Table 1. Radio power characterization

Radio mode	Power consumption (mW)
Transmit (T_x)	14.88
Receive (R_x)	12.50
Idle	12.36
Sleep	0.016

At time t_1, the node wants to wake up one of its neighbors and thus becomes an initiator. It starts sending beacon packets on frequency f_1, until it receives a response from the target node, which happens at time t_2. At this moment, the radio in frequency band f_2 is turned on for regular data transmissions. Note that at the same time, the radio in band f_1 still wakes up periodically from its sleep state to listen if any nodes want to contact it. After the data transmissions have ended (e.g. at the end of a predetermined stream of packets, after a timeout, etc.), the node turns its radio in band f_2 off again. At time t_4, it receives a beacon from another initiator node while listing in the f_1 band. The node responds to the initiator and turns its radio on again in band f_2.

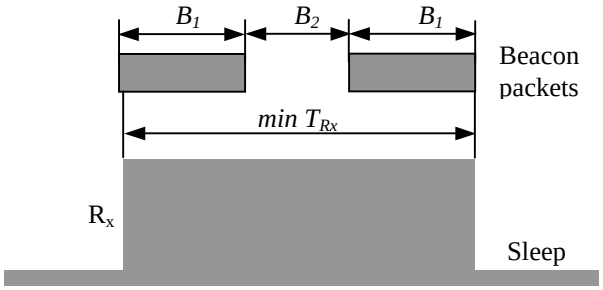


Figure 3 – Radio on-time in the wakeup plane

In order for the target node to receive at least one beacon, it needs to turn on its radio for a sufficiently long time, denoted as T_{Rx}. Figure 3 illustrates the worst-case situation where the radio is turned on just too late to receive the first beacon. In order to receive the second beacon, T_{Rx} should be at least as long as twice the transmit time B_1 of a beacon packet, plus the inter-beacon spacing B_2 that is required to allow the target node to respond.

3. THEORETICAL ANALYSIS

Setup Latency

Before simulating our protocol, we first develop a theoretical model of the system performance. We define the ***setup latency T_S of a link*** *as the interval from the time the initiator starts sending out beacons, to the time the target node has responded to the beacon.* Typically the target and originator node are not synchronized, which means that the beacon sending process starts at a random point in the cycle of the target node. As a result, the start of the first beacon is distributed uniformly random in interval T. Figure 4 shows the values of T_S, normalized versus $B_{1+2} = B_1 + B_2$, for different start times of the beacon sending process.

It is clear that T_S only takes on integer multiples of B_{1+2}, as this is the time it takes to send a beacon and receive the response to it. For the region that is labeled i in Figure 4, the setup latency is equal to $i \cdot B_{1+2}$, since beacon i is the first one to fall entirely within the interval of length T_{Rx} when the target node's radio is on. The probability of being in region i is equal to the length of that region divided by T. As a result, for $T > T_{Rx}$, the statistics of T_S can be derived from Figure 4 as:

$$\begin{cases} P(T_S = B_{1+2}) = \dfrac{T_{Rx} - B_1}{T} \\ P(T_S = k \bullet B_{1+2}) = \dfrac{B_{1+2}}{T} \qquad k = 2...K \\ P(T_S = (K+1) \bullet B_{1+2}) = \dfrac{T - K \bullet B_{1+2} - T_{Rx} + B_1 + B_{1+2}}{T} \end{cases} \quad (1)$$

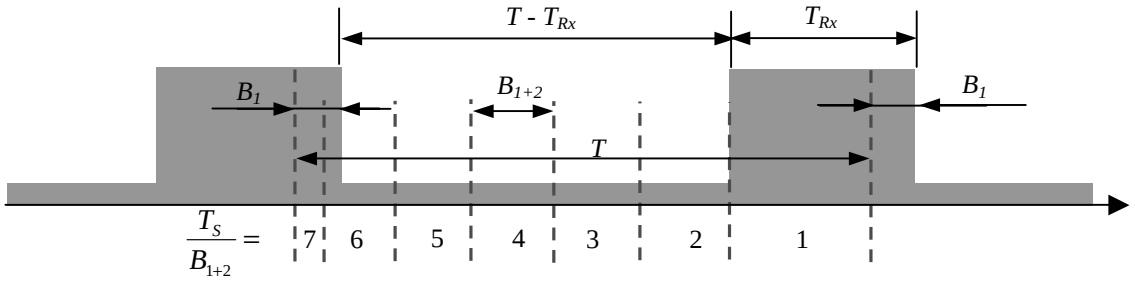


Figure 4 – Analysis of the setup latency

$$K = \left\lfloor \frac{T - T_{Rx} + B_1 + B_{1+2}}{B_{1+2}} \right\rfloor \quad (2)$$

Based on these equations, we calculate the average setup latency $\overline{T}_S$ for a link. To simplify the expressions, we select T_{Rx} equal to its minimum value (see Figure 3):

$$T_{Rx} = B_{1+2} + B_1 \quad (3)$$

In this case, (1)-(2) reduce to:

$$\begin{cases} P(T_S = k \bullet B_{1+2}) = \dfrac{B_{1+2}}{T} & k = 1...K \\ P(T_S = (K+1) \bullet B_{1+2}) = \dfrac{T - K \bullet B_{1+2}}{T} \end{cases} \quad (4)$$

$$K = \left\lfloor \frac{T}{B_{1+2}} \right\rfloor = \frac{T}{B_{1+2}} - \delta \quad (5)$$

The average setup latency per hop can be derived from (4) as being equal to (6), where δ is defined in (5).

$$\overline{T}_S = \frac{T + B_{1+2}}{2} + \frac{B_{1+2}^2}{2 \bullet T} \bullet \delta \bullet (1 - \delta) \quad (6)$$

If T is an integer multiple of B_{1+2}, this expression simplifies to:

$$\overline{T}_S = \frac{T + B_{1+2}}{2} \quad (7)$$

Equations (6) and (7) are valid on condition that $T > T_{Rx}$. For the special case when there is no sleep period, $T = T_{Rx}$ and the average setup delay is equal to:

$$\overline{T}_S = B_{1+2} \quad (8)$$

Energy Savings

The total energy consumed by a node during a time interval t can be broken up into two components, one for each frequency band.

$$E_{node} = E_{wakeup} + E_{transfer} \quad (9)$$

Equation (10) details the energy consumption in the wakeup plane. The first term accounts for the listening cycle, where P_{node} is given by (11). In this equation P^0_{node} is a combination of idle and receive power. Since both are very similar, see Table 1, we can approximate P^0_{node} by P_{idle}. The second term in (10) represents the energy consumption of transmitting beacon and response packets (P_{setup} is thus a combination of transmit, receive and idle power).

$$E_{wakeup} = P_{node} \bullet (t - t_{setup}) + P_{setup} \bullet t_{setup} \quad (10)$$

$$P_{node} = \frac{P_{sleep} \bullet (T - T_{Rx}) + P^0_{node} \bullet T_{Rx}}{T} \quad (11)$$

The energy consumption in the transfer plane is given by (12). In this equation, t_{data} is the total time the radio is turned on in the transfer plane for communicating data. As a result, P_{data} contains contributions of packet transmission, packet reception and idle power.

$$E_{transfer} = P_{sleep} \bullet (t - t_{data}) + P_{data} \bullet t_{data} \quad (12)$$

Without topology management, the total energy would be equal to (13). Although P_{data} also contains contributions of P_{idle}, we have chosen to split up the energy consumption in analogy with (12) for ease of comparison. The main difference is that the radio is never in the energy-efficient sleep state here.

$$E^{original}_{node} = P_{idle} \bullet (t - t_{data}) + P_{data} \bullet t_{data} \quad (13)$$

The gain in terms of energy obtained by using STEM is the difference between (13) and (9):

$$\begin{aligned} . E_{node} &= E^{original}_{node} - E_{node} \\ &= (P_{idle} - P_{sleep} - P_{node\bullet}) \bullet t - (P_{idle} - P_{sleep}) \bullet t_{data} \\ &\quad - (P_{setup} - P_{node\bullet}) \bullet t_{setup} \end{aligned} \quad (14)$$

Since we consider scenarios where the node is in the monitoring state most of the time, we can roughly disregard t_{data} and t_{setup}. By ignoring the minute power of the sleep state and substituting P^0_{node} in (11) by P_{idle}, we approximate (14) as:

$$. E_{node} \quad P_{idle} \bullet t \bullet (1 - \frac{T_{Rx}}{T}) \quad (15)$$

Furthermore, by also ignoring t_{data} in (13), we can reasonably approximate the relative gain in terms of energy as:

$$. E_{node} = \frac{. E_{node}}{E^{original}_{node}} \quad 1 - \frac{T_{Rx}}{T} \quad (16)$$

From (6) and (16), we can derive the general relationship between the setup latency and the relative energy gain for a node. For the special case where T is an integer multiple of B_{1+2}, as in (7), this relationship is given by:

$$. E_{node} = 1 - \frac{T_{Rx}}{\left(2 \bullet \overline{T}_S - B_{1+2}\right)} \quad (17)$$

Since the node has a finite battery capacity, these energy savings directly correspond to the same relative increase in the lifetime of a node, which ultimately results in a prolonged lifetime of the sensor network.

4. PERFORMANCE EVALUATION

Simulation Setup

In this section, we verify our algorithm through simulations, which were written on the Parsec platform, an event-driven parallel simulation language [12]. We distribute N nodes randomly over a square field of size L x L and each of them has a transmission range R.

For a uniform network density, the probability $Q(n)$ for a node to have n neighbors in a network of N nodes is given by the binomial distribution of (18), when edge effects are ignored. In this equation, Q_R is the probability of a node being in the transmission range of a particular node, given by (19). We use the symbol Q in this paper for probabilities, to avoid confusion with power (denoted by P).

$$Q(n) = Q_R^n \bullet (1 - Q_R)^{N-1-n} \bullet \binom{N-1}{n} \tag{18}$$

$$Q_R = \frac{\pi R^2}{L^2} \tag{19}$$

For large values of N, tending to infinity, this binomial distribution converges towards the Poisson distribution (20) [13]. The network connectivity is thus only a function of the average number of neighbors of a node, denoted by parameter . .

$$Q(n) = \frac{\cdot^n}{n!} \bullet e^{-\cdot} \tag{20}$$

$$\cdot = \frac{N}{L^2} \bullet \pi R^2 \tag{21}$$

Since the traffic communication patterns depend solely on the network connectivity, we only have to consider . and not N, R and L separately. We have verified this statement through simulations, and therefore can characterize a uniform network density by the single parameter . .

In our simulations, we have chosen R = 20 m, which corresponds to the numbers in Table 1. The area of the sensor network is such that for N = 100, we have . = 20. Furthermore, our setup includes a CSMA-type MAC, similar to the DCF of 802.11. Table 2 lists the other simulation settings, where L_{beacon} and $L_{response}$ are the sizes (including MAC and PHY header) of the beacon and the response packets respectively.

Table 2. Simulation settings

R	20 m	R_b	2.4 Kbps
L	79.27 m	B_{1+2}	150 ms
L_{beacon}	144 bits	T_{Rx}	225 ms
$L_{response}$	144 bits		

The node closest to the top left corner detects an event and sends 20 information packets of 1040 bits to the data sink with an inter-packet spacing of 16 seconds. This process will therefore take about t^* = 320 seconds. The data sink is the sensor node located closest to the bottom right corner of the field. We have observed that the average path length is between 6 and 7 hops. All reported results are averaged over 100 simulation runs.

Simulation Results

Figure 5 shows the average setup latency per hop as a function of the wakeup period T. The dashed curve with the markers is obtained via simulations, while the top solid curve corresponds to (6). There is a constant offset, which is due to the fact that the transmission time of a beacon and response packet is actually 120 ms, while the beacon period B_{1+2} was chosen conservatively to be 150 ms. The actual setup latency is thus comprised of a number of B_{1+2} periods, plus the time to transmit a beacon and receive the response, which is about 30 ms less than what is calculated theoretically in (6). From Figure 5, we observe that if we correct (6) by subtracting 30 ms, the correspondence to simulations is indeed very close.

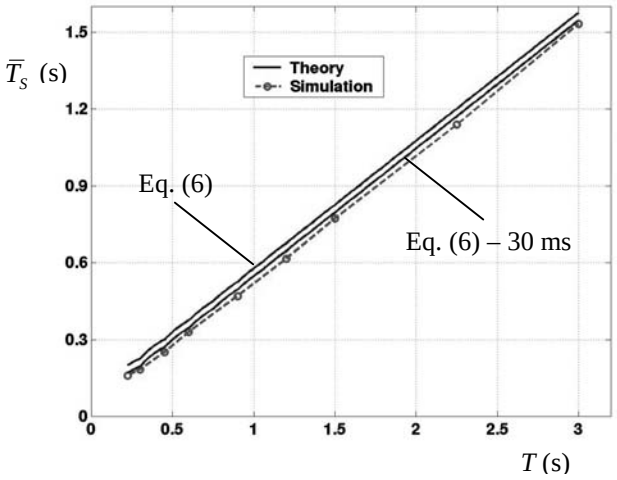


Figure 5 – Average setup latency

In Figure 6, the total energy is plotted versus the normalized observation interval t/t^*. As a basis for comparison, we included the curve for a scheme without topology management, which corresponds to (13). In this case, there is only one radio, which can never be turned off. The other dashed curves represent the performance for STEM with different values of T. The theoretical results, plotted using solid lines, are obtained by multiplying the curve without topology management by (1-. E), see (16).

For all values of t, the same number of packets is sent, meaning that the duration of the transfer state is kept constant, and is approximately equal to t^*. When t increases,

the monitoring state becomes more predominant. As a result, t_{data} and t_{setup} in (10), (12), (14) are negligible for large t, such that the simulated values start approaching the theoretical ones. We observe that STEM results in energy savings when $t > 2 \cdot t^*$, which means that the network should reside in the monitoring state 50% or more of the time.

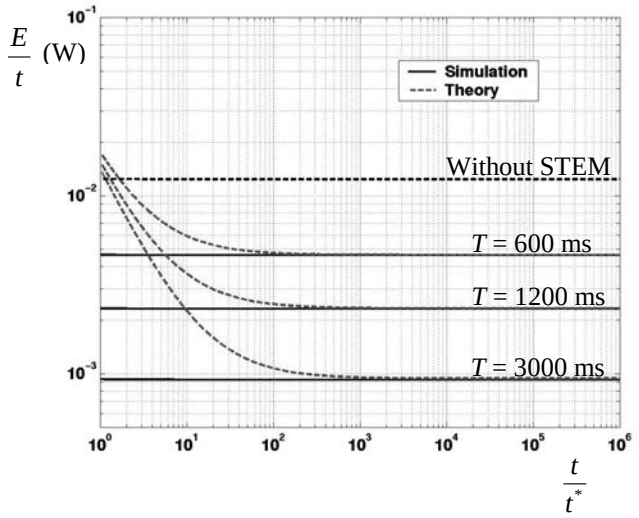


Figure 6 – Relative energy savings versus the total observation interval t

Figure 7 explicitly shows the tradeoff between energy savings and setup latency. The solid theoretical curves are obtained from (16) and (6), with and without the correction that was introduced in Figure 5. We have plotted the simulated results for values of the different observation period t.

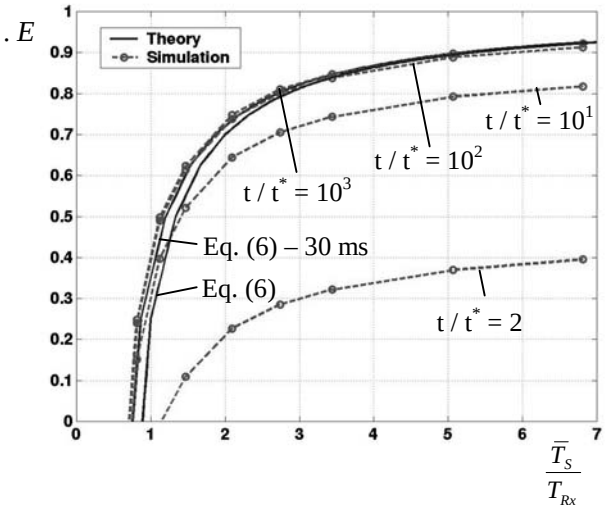


Figure 7 – Simulated energy – setup latency tradeoff

For large t, the simulated performance converges to the theoretical one. This corresponds to a regime where the monitoring state heavily dominates the transfer state, which is the focus of this work. We note, however, that STEM can also be valuable if outside this regime (*i.e.*, for smaller values of t), although the gains are much less pronounced in this case.

5. COMBINING STEM AND GAF

As mentioned in the introduction, existing topology management schemes, such as GAF and SPAN, coordinate the radio sleep and wakeup cycles while ensuring adequate communication capacity. STEM can be viewed as being orthogonal to these schemes, and additional gain is achieved by considering combinations STEM-GAF or STEM-SPAN. In this work, we specifically focus on the interaction between STEM and GAF.

GAF Behavior

In this subsection, we discuss plain GAF, *i.e.*, without STEM. The GAF algorithm is based on a division of the sensor network in a number of virtual grids of size r by r, see Figure 8. The value of r is chosen such that all nodes in a grid are equivalent from a routing perspective [6]. This means that any two nodes in adjacent grids should be able to communicate with each other. By investigating the worst-case node locations depicted in Figure 8, we can calculate that r should satisfy (22) [6].

$$r = \frac{R}{\sqrt{5}} \tag{22}$$

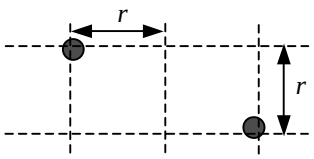


Figure 8 – GAF grid structure

The average number of nodes in a grid, M, is given by (23). By combining this with (22), we can see that M should satisfy (24). The average number of nodes in a grid is thus fairly low. Even if r satisfies (22) with equality, which we assume to hold for the remainder of this paper, M is smaller than 2 for densities of $\cdot = 31$. To put this into perspective, $\cdot = 31$ corresponds to a topology where each node has 31 neighbors on average.

$$M = \frac{N}{L^2} \bullet r^2 \tag{23}$$

$$M = \frac{\cdot}{5\pi} \tag{24}$$

Since all nodes in a grid are equivalent from a routing perspective, we can use this redundancy to increase the

network lifetime. GAF only keeps one node awake in each grid, while the other nodes put their radio in the sleep mode. To balance out the energy consumption, the burden of traffic forwarding is rotated between nodes. For simplicity, we ignore the unavoidable time overlap of this process associated with handoff. If there are *m* nodes in a grid, the node will (ideally) only turn its radio on *1/m*th of the time and therefore will last *m* times longer. However, equation (24) shows that the redundancy is rather low on average, even for fairly dense networks.

When distributing nodes over the sensor field, some grids will not contain any nodes at all. We use . to denote the fraction of used grids, *i.e.*, which have at least one node. As a result, the average number of nodes in the used grids is equal to *M'*, given by:

$$M \ni = \frac{M}{.} \tag{25}$$

The average power consumption of a node using GAF, $\overline{P}_{node}^{GAF}$, is equal to (26). In this equation, P_{on} is the power consumption of a node if GAF would not be used. It thus contains contributions of receive, idle and transmit mode, as the node would never turn its radio off. With GAF, in each grid only one node at a time has its radio turned on, so the total power consumption of a grid, P_{grid}, is virtually equal to P_{on} (neglecting the sleep power of the nodes that have their radio turned off). Since *M'* nodes share the duties in a grid equally, the power consumption of a node is *1/M'* that of the grid, as in (26).

$$\overline{P}_{node}^{GAF} = \frac{P_{on}}{M \ni} = \frac{P_{grid}}{M \ni} \tag{26}$$

The average relative gain in energy for a node is thus given by:

$$. E_{node} = \frac{P_{on} \bullet t - P_{node}^{GAF} \bullet t}{P_{on} \bullet t} = 1 - \frac{1}{M \ni} \tag{27}$$

Alternatively, we see that the lifetime of each node in the grid is increased with the same factor *M'*. As a result, the average lifetime of a grid, $\bar{t}_{grid}$, *i.e.*, the time that at least one node in the grid is still alive, is given by (28), where t_{node} is the lifetime of a node without GAF. ***We can essentially view a grid as being a 'virtual node', composed of M' actual nodes.***

$$\bar{t}_{grid} = t_{node} \bullet M \ni \tag{28}$$

Note that $\overline{P}_{node}^{GAF}$ and $\bar{t}_{grid}$, which are averages over all grids, only depend on *M'* and not on the exact distribution of nodes in the used grids! Of course, the variance of both the node power and the grid lifetime depends on the distribution. If we would have full control over the network deployment, we could make sure that every used grid has exactly *M'* nodes, which minimizes the power and lifetime variance.

For the special case of a random node distribution, we now calculate the statistics exactly. The probability *Q*(*m*) of having a grid with *m* nodes is given by (29). The derivation is analogous that the one leading to (20).

$$Q(m) = \frac{M^m}{m!} \bullet e^{-M} \tag{29}$$

In this case, the fraction . of used grids is equal to:

$$. = 1 - Q(0) = 1 - e^{-M} \tag{30}$$

The probability of having *m* nodes in a used grid is given by:

$$Q(m \mid m = 1) = \frac{Q(m)}{Q(m = 1)} = \frac{M^m}{m!} \bullet \frac{e^{-M}}{1 - e^{-M}} \tag{31}$$

We also know that the probability that power of a node is equal to *1/m*th of that in a grid, is the same as the probability of a node being in a grid with *m* nodes:

$$Q(P_{node}^{GAF} = \frac{P_{grid}}{m}) = \frac{m \bullet Q(m)}{M} = \frac{M^{m-1}}{(m-1)!} \bullet e^{-M} \tag{32}$$

Alternatively, equation (33) gives the probability that the lifetime of a grid is *m* times that of an individual node.

$$Q(t_{grid} = t_{node} \bullet m) = Q(m \mid m = 1) = \frac{M^m}{m!} \bullet \frac{e^{-M}}{1 - e^{-M}} \tag{33}$$

We can verify from (32) and (33) that the average values of P_{node}^{GAF} and t_{grid} are indeed equal to (26) and (27) respectively.

Interaction of STEM and GAF

As mentioned before, GAF essentially places one virtual node in each grid, and the physical nodes alternatively perform the functionalities of that virtual node. From this perspective, combining GAF with STEM is straightforward by envisioning the virtual node as running STEM. In real life, nodes alternate between sleep and active states, as governed by GAF. The one active node in the grid, runs STEM in the same way as described in section 2. The only difference is that now the routing protocol needs to address virtual nodes (or grids) instead of real nodes.

This insight allows us to directly modify the expressions of section 3 to similar ones for the combination of STEM-GAF. In particular, (16) becomes (34), where the statistics of *m* are given by (32).

$$. E_{node} = 1 - \frac{1}{m} \bullet \frac{T_{Rx}}{T} \tag{34}$$

When considering the average behavior over all grids, we get:

$$. E_{node} = 1 - \frac{1}{M} \cdot \frac{T_{Rx}}{T} \quad (35)$$

If T is an integer multiple of B_{1+2}, we can combine (35) with (7) to obtain the following tradeoff between energy savings and setup latency:

$$. E_{node} = 1 - \frac{1}{M} \cdot \frac{T_{Rx}}{\left(2 \cdot \overline{T}_S - B_{1+2}\right)} \quad (36)$$

Figure 9 plots this tradeoff for different values of M'. As argued before, these curves are independent of the exact node distribution, but only depend on M'.

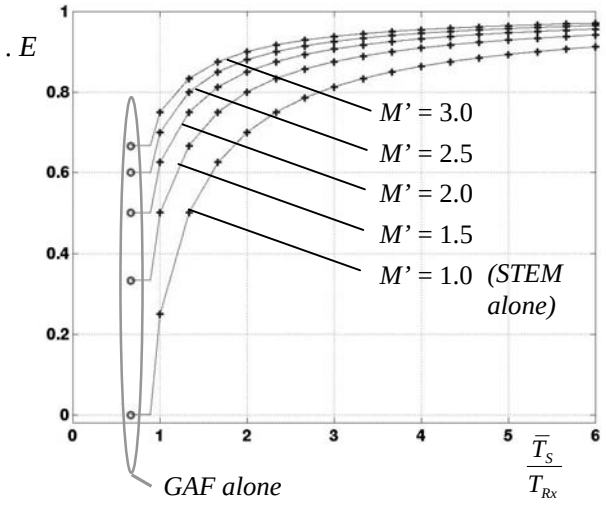


Figure 9 – Theoretical energy – setup delay tradeoff

The solid curves are based on (6) and (35). They therefore represent the behavior as averaged over the different grids, for any $T > T_{Rx}$. On these curves, the '+' markers are points obtained from (36), where T is an integer multiple of B_{1+2}.

The circles mark the limiting case where the wakeup-plane radio is always on ($T = T_{Rx}$). This case corresponds to a traditional paging channel setup, where a separate paging radio is used to wake up the main data radio [10]. Of course, this only makes sense if the paging radio is substantially more energy efficient than the main one. By looking at (27) and (35), we notice that in this case the energy savings are also the same as those of pure GAF, without STEM, which corresponds to intuition. The circles can therefore be viewed as the (energy) behavior of GAF as well, although, strictly speaking, the setup latency does not have any true meaning here.

By comparing (16) and (35), we note that the curve with M' = 1 can also be viewed as representing the case of STEM without GAF, where essentially no node redundancy is exploited. So, besides the combination of STEM-GAF, figure 9 also shows the behavior of GAF without STEM (circles) and of STEM without GAF (curve with M' = 1).

We notice that by allowing more setup latency, the energy savings can be increased considerably beyond what is achievable by GAF alone. For a uniform node deployment, the values of M' in Figure 9 translate to M and . as given by (21), (23) and (25). Table 3 lists the values of these parameters for the curves of Figure 9.

Table 3. Density mapping for a uniform node distribution

M'	M	.
1.0	0	0
1.5	0.87	13.7
2.0	1.59	25.0
2.5	2.22	35.0
3.0	2.82	44.3

Note that for moderate node densities (. < 25), the average redundancy in a used grid is fairly low. As a result, GAF alone only results in moderate energy saving, below 34%. On the other hand, by incorporating STEM, we can achieve savings of more than 93%! In other words, the energy is reduced to 66% of the original value by GAF and to a mere 7% by also using STEM. The penalty is of course an increased setup delay.

6. CONCLUSIONS

In this paper, we have introduced STEM, a topology management technique that trades off power savings versus path setup latency in sensor networks. It emulates a paging channel by having a separate radio operating at a lower duty cycle. Upon receiving a wakeup message, it turns on the primary radio, which takes care of the regular data transmissions.

Our topology management is specifically geared towards those scenarios where the network spends most of its time waiting for events to happen, without forwarding traffic. STEM leverages the fact that, while awaiting events, the network capacity can be heavily reduced, resulting in energy savings.

We have shown that STEM integrates directly with other topology management schemes such as GAF, and results in energy savings above and beyond these existing techniques. Compared to a network without topology management, a combination of GAF and STEM can reduce the energy consumption to a mere 7%. Alternatively, this results in a node lifetime increase of a factor 14! However, these

benefits come at the cost of increased setup latency, which is linearly proportional to the number of hops in the multi-hop path. It will depend on the specific applications, how much latency is allowed, and therefore how far the energy consumption can be scaled down.

Analyzing the interaction of STEM and SPAN is a topic of future research. Another issue worth investigating is how power control strategies can be incorporated into topology management.

7. ACKNOWLEDGEMENTS

We would like to thank Sachin Adlakha for his useful feedback, especially the theoretical analysis section.
This paper is based in part on research funded by the NSF CAREER award and by the DARPA Power Aware Computing and Communications (PAC/C) program through AFRL contract # F30602-00-C-0154. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF, DARPA, Air Force Rome Laboratory or the U.S. Government.

REFERENCES

[1] K. Sohrabi, J. Gao, V. Ailawadhi, G. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications Magazine*, Vol.7, No.5, pp. 16-27, Oct. 2000.

[2] L. Clare, G. Pottie, J. Agre, "Self-organizing distributed sensor networks," *SPIE - The International Society for Optical Engineering*, Orlando, FL, pp. 229-237, April 1999.

[3] D. Estrin, R. Govindan, "Next century challenges: scalable coordination in sensor networks," *MobiCom 1999*, Seattle, WA, pp. 263-270, August 1999.

[4] A. Savvides, C.-C. Han, M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," *MobiCom 2001*, Rome, Italy, pp. 166 – 179, July 2001.

[5] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *MobiCom 2001*, Rome, Italy, pp. 70-84, July 2001.

[6] Y. Xu, J. Heidemann, D. Estrin, "Geography-informed energy conservation for ad hoc routing," *MobiCom 2001*, Rome, Italy, pp. 70-84, July 2001.

[7] J.-H. Chang, L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," *INFOCOM 2000*, Tel Aviv, Israel, pp. 22-31,March 2000.

[8] J. Rabaey, J. Ammer, J.L. da Silva, D. Patel, "PicoRadio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes," *IEEE Computer Society Workshop on VLSI 2000*, Orlando, FL, pp. 9-12, April 2000.

[9] W. Rabiner Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *HICSS 2000*, Maui, HI, Jan. 2000.

[10] S. Kumar, "DARPA sensor information technology (SensIT)," *http://www.darpa.mil/ito/research/sensit/*.

[11] Sensoria Corporation, *http://www.sensoria.com/*.

[12] R. Bagrodia, R. Meyer, M. Takai, Y.A. Chan, X. Zeng, J. Marting, H.Y. Song, "Parsec: a parallel simulation environment for complex systems," *Computer*, Vol.31, No.10, pp. 77-85, October 1998.

[13] M. Yacoub, *Foundations of Mobile Radio Engineering*, CRC Press, 1993.

Curt Schurgers *received his MSEE summa cum laude from the Katholieke Universiteit Leuven (KUL), Belgium, in 1997. From 1997 to 1999, he worked at IMEC (Interuniversity Micro Electronics Center, Leuven, Belgium) on memory optimization techniques for turbo codes. Currently, he is pursuing his Ph.D. degree UCLA, focusing on energy efficient communication and networking systems. Curt Schurgers has received the F.W.O., the B.A.E.F. and the UCLA fellowships in 1997, 1999 and 2000 respectively.*

Vlasios Tsiatsis *has received his BS degree from the Technical University of Crete, Chania, Greece in 1998, and his M.S. degree in EE from the University of California, Los Angeles (UCLA), in 2001. He is currently pursuing his Ph.D. degree at UCLA, researching low power protocol architectures. In 1998 and 2000, he was honored with the UCLA fellowship.*

Mani Srivastava *is an Associate Professor of Electrical Engineering at UCLA. He received his B.Tech. in EE from IIT Kanpur in India and M.S. and Ph.D. from Berkeley. From 1992 through 1996 he was a Member of Technical Staff at Bell Laboratories in Networked Computing Research. His current research interests are in mobile and wireless networked computing systems, low power systems, and sensor networks. He received the NSF CAREER award in 1997, and the President of India Gold Medal in 1985.*

Appendix I:

Optimizing Sensor Networks in the Energy-Latency-Density Design Space

Curt Schurgers, *Student Member*, *IEEE*, Vlasios Tsiatsis,
Saurabh Ganeriwal, and Mani Srivastava, *Member*, *IEEE*

Abstract—In wireless sensor networks, energy efficiency is crucial to achieving satisfactory network lifetime. To reduce the energy consumption significantly, a node should turn off its radio most of the time, except when it has to participate in data forwarding. We propose a new technique, called Sparse Topology and Energy Management (STEM), which efficiently wakes up nodes from a deep sleep state without the need for an ultra low-power radio. The designer can trade the energy efficiency of this sleep state for the latency associated with waking up the node. In addition, we integrate STEM with approaches that also leverage excess network density. We show that our hybrid wakeup scheme results in energy savings of over two orders of magnitude compared to sensor networks without topology management. Furthermore, the network designer is offered full flexibility in exploiting the energy-latency-density design space by selecting the appropriate parameter settings of our protocol.

Index Terms—Sensor networks, energy efficiency, wakeup, topology.

1 Introduction

Wireless sensor networks are autonomous ad hoc networks designed for monitoring tasks, such as battlefield surveillance, equipment supervision, intruder detection, and wildlife observation, among many others [1], [2], [3]. Sensor networks are made up of a large number of tiny devices, called sensor nodes, which contain integrated sensors, processors, and radios. The nodes gather various sensor readings, process them, coordinate among each other, and forward the processed information to a data sink. This forwarding typically occurs wirelessly via other nodes using a multihop path [3], [9]. The crucial design challenge in sensor networks is energy efficiency as the individual nodes have only a small battery as a power source. To achieve satisfactory network lifetime, energy efficiency is really a problem that needs to be tackled on the level of the entire network. One of the key aspects is the organization of network communications as the radio is the main energy consumer in a sensor node [1], [3], [4]. The only way to reduce this energy is to completely turn the radio off [4]. However, besides sensing their environment, nodes also form the ad hoc network needed to forward the data to the data sink. Topology management schemes coordinate which nodes turn their radio off and when such that the traffic forwarding remains satisfactory while minimizing the network energy consumption.

Consider, for example, a sensor network that is designed to detect brush fires. It has to remain operational for months or years while only sensing if a fire has started. Once a fire is detected, this information should be forwarded to the user quickly. Even when we want to track how the fire spreads, it probably suffices for the network to remain up only for an additional week or so. Similar observations hold for applications such as surveillance of battlefields, machine failures, room occupancy, or other reactive scenarios, where the user needs to be informed once a condition is satisfied. The majority of the time, the network is only sensing its environment, which we refer to as the network being in the *monitoring state*. Once an event happens, data needs to be forwarded to the data sink and the network transitions to the more active *transfer state*.

We propose a new topology management scheme, called ***STEM (Sparse Topology and Energy Management)***. It reduces the energy consumption in the monitoring state to a bare minimum while ensuring satisfactory latency for transitioning to the transfer state. In fact, STEM allows us to efficiently trade one design constraint (energy) for the other (latency). We also combine it with techniques that leverage increased network density to obtain energy savings. In essence, our scheme thus offers the designers full flexibility in trading latency, density, and energy versus each other.

Furthermore, we derive the mathematical model that governs these tradeoffs. For example, for a specific desired network lifetime and acceptable notification latency, we can calculate the network density that is required. When STEM is running in the network, this density will assure that both the latency and the network lifetime are as desired. This model is therefore a tool for the network designer to choose the optimal parameter settings given the deployment requirements. At design time, it allows the selection of the desired operating point in the latency-density-lifetime design space.

2 Related Work

For routing in sensor networks, two alternative approaches have been considered: flat multihop and clustering.

• *The authors are with the Networked and Embedded Systems Lab (NESL), Electrical Engineering Department, University of California at Los Angeles, 56-125B Eng. IV, UCLA-EE Dept., Los Angeles, CA 90095. E-mail: {curts, tsiatsis, saurabh, mbs}@ee.ucla.edu.*

Manuscript received 7 Mar. 2002; revised 13 May 2002; accepted 13 May 2002.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number 7-032002.

TABLE 1
Radio Power Characterization

Radio mode	Power consumption (mW)
Transmit (T_x)	14.88
Receive (R_x)	12.50
Idle	12.36
Off	0.016

Although STEM is applicable to both of them, we mainly focus on flat multihop routing [3], [7], [8]. For clustered approaches [9], which are possibly hierarchical, our scheme can be used to reduce the energy of the cluster heads. Recently, topology management techniques, called SPAN [5] and GAF [6], have been proposed for flat multihop routing. They trade network density for energy savings while preserving the data forwarding capacity of the network. However, the absence of traffic in the monitoring state is not exploited at all. By integrating these schemes into STEM, as we describe in Section 6, we can combine their benefits with those of STEM to achieve compounded energy savings.

STEM is essentially a technique to quickly transition to the transfer state, while making the monitoring state as energy efficient as possible. Other authors have proposed to do this wake up using a separate paging channel [10]. This approach critically assumes that the listen mode of this paging radio is ultra low power. However, the difference in the transmission range between the data and wakeup radio presents a major difficulty. STEM offers an alternative by trading energy for latency. Furthermore, if such a low-power radio is available, the energy savings are further improved by using it in a low duty cycle, as STEM does. The work in [14] describes an algorithm that also uses a low duty cycle radio. This algorithm is designed for a different goal, namely, to discover the network topology some time after its deployment. It is less aggressive than STEM and, therefore, would result in much higher latencies to transition to the transfer state. The same principle of duty cycling the radio is also adapted in the Medium Access Control (MAC) protocol presented in [16], called S-MAC. However, channel access and node wakeup are integrated together. In the monitoring state, where there is no data to forward, STEM is therefore more energy efficient than S-MAC while assuring timely transitioning to the transfer state. In this transfer state, our approach allows for any MAC protocol, including S-MAC.

3 Sparse Topology and Energy Management

3.1 Basic Concept

In the monitoring state, where there is no traffic to forward, only the node's sensors and some preprocessing circuitry are on. For simplicity, we refer to this state as the node being asleep or in the sleep state. When a possible event is detected, the main processor is awakened to analyze the data in more detail. The radio is only turned on if the processor decides that the information needs to be communicated to other nodes. The reason for this can be understood from the radio mode power numbers in Table 1.

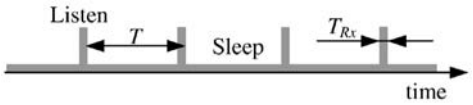


Fig. 1. Low-power listen mode.

These are for the TR1000 radio from RF Monolithics [15], where the transmit range is set to approximately 20 meters [4]. This low-power radio has a data rate of 2.4 Kbps and uses On-Off Keying (OOK [18]) modulation. As can be observed from this table, the radio consumes considerable power except when completely turned off.

To forward traffic, nodes on the multihop path need to be awakened, or, equivalently, transition from the monitoring to the transfer state. The problem is that these nodes have no way of knowing when to transition if they did not detect that same event. Thus, the dilemma we are faced with is the following: For energy reasons, the nodes should turn off their radio when in the monitoring state, but still need to be told somehow if they should turn it back on. As a solution, each node periodically turns on its radio for a short time to listen if someone wants to communicate with it. In the monitoring state, instead of complete being asleep, a node goes into this low-power listen mode, as shown in Fig. 1. The period of the listen-sleep cycle is denoted as T. The node that wants to communicate, the *initiator node*, polls the node it is trying to wake up, called the *target node*. We will detail the nature of these polls in Section 3.3. As soon as the target node, which is in the low-power listen mode of Fig. 1, hears the poll, the link between the two nodes is activated. If the packet needs to be relayed further, the target node will become an initiator for the next hop and the process is repeated.

Once the link between nodes is activated, data is transferred using a MAC protocol. This MAC protocol is only used in the transfer state as even the most efficient one would consume a lot more energy than our low-power listen mode. The reason is that MAC protocols are designed to organize access to the shared medium, in addition to contacting nodes. Our strategy is thus to decouple the transfer and wakeup functionalities: In the monitoring state, we consume as little energy as possible and only in the transfer state is the MAC protocol started.

3.2 Alternative Setups

Without special protocol provisions, nodes are not synchronized and, therefore, do not know the phase of each other's wakeup-sleep cycles in the listen mode. To avoid missing the short time the target node has its radio on, the initiator has to poll continuously. As the data arrivals are uncorrelated with the sleep cycles, it will take about half a cycle for the target to hear the poll. However, this aggressive polling causes problems, as shown in Fig. 2. A

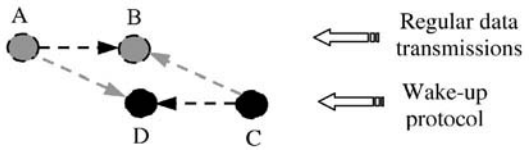


Fig. 2. Interference problem of aggressive wakeup.

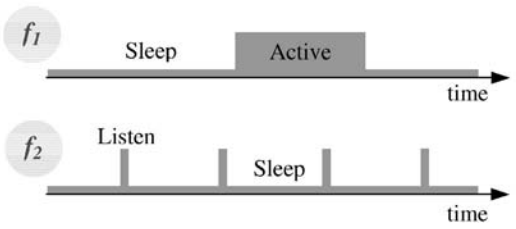


Fig. 3. Separate data and wakeup using two radios.

regular data transmission is going on between nodes A and B. When node C wants to wake up D, its aggressive polls will collide with the ongoing data transmission, essentially acting as a jammer to B. Despite possible recovery action from the MAC, the data communication between A and B suffers from extra delays. As we will illustrate in Section 4, more energy can be saved if we allow more time to set up a link between two nodes. Therefore, we might desire this setup procedure to be relatively long, but its impact is only felt at the start of a communication epoch. However, it is typically undesirable as this setup would also cause equally long disruptions of ongoing transmissions.

Since this aggressive nature is needed to limit the wakeup latency, the solution is to completely separate data transfer from wakeup. A natural choice is to use two radios operating in separate frequency bands. As shown in Fig. 3, the radio in band f_1 is only turned on in the transfer state, and the wakeup band f_2 can be viewed as a separate paging channel. Unlike [10], we are not limited by the availability of an ultra low power radio for this paging channel. Instead, we can use the most efficient radio available and further reduce the energy consumption by putting it in our low-power listen mode. This allows us to trade energy savings versus latency, beyond the capabilities of the radio alone, as we will detail later on.

In principle, we could use one radio and let it switch between frequencies. However, if a target node is already transferring data and also has to wake up another node, it has to interrupt its data transmission or postpone the wakeup. Both are undesirable and, therefore, we choose to use two radios. The penalty of making the node more expensive is minimal, as the radio typically accounts for less than 15 percent of the cost of a sensor node (e.g., an extra TR1000 [15] radio on the MICA motes [17]).

An alternative way to separate data transmissions from wakeup is to assign them to separate time slots, as shown in Fig. 4. In this case, the initiator only sends the poll in the slot to which the target is listening. Unfortunately, this option requires time synchronization as all nodes have to remain synchronized at all times, which brings about considerable overhead. The monitoring state would thus be much more energy hungry than in the two radio option of Fig. 3. As we will explain in Section 6, the same energy savings as in the setup of Fig. 3 could be achieved by deploying more nodes, but the total network deployment cost would exceed that of using slightly more expensive nodes. We therefore opt for the solution with two radios. Sensor nodes developed by Sensoria Corporation [11], for example, are already equipped with a dual radio.

Fig. 4. Separate data and wakeup in time.

3.3 Operation of STEM-B and STEM-T

To poll the target node, the initiator sends a stream of beacon packets in band f_2. Each beacon contains the MAC address of both the target and initiator node. As soon as the target receives a beacon, it turns on its data radio in band f_1 and also sends back an acknowledgment in band f_2. This way, the initiator knows when it can stop polling, resulting in a reduction in setup latency, as we will detail in Section 4. The length of the interbeacon interval T_B is such that there is sufficient time to send the beacon and receive the acknowledgment. The time during which the radio is turned on in the listen mode is denoted as T_{Rx} (see Fig. 1). In order to guarantee that the target node receives at least one beacon, T_{Rx} needs to be at least as long as the transmit time of a beacon plus the interbeacon interval T_B.

It is still possible that collisions between beacons occur. To handle this issue, a node also turns on its data radio if it hears a collision during the listen interval T_{Rx}. A collision can be detected by monitoring the RSSI (received signal strength indicator) of the radio. In this case, the node does not send back an acknowledgment as it would likely collide with that of other nodes that are also awakened this way. After transmitting the beacon stream for a sufficient amount of time (approximately equal to T; we derive the exact expression in Section 4), the initiator node can be sure that the target node has turned on its data radio in band f_1. Indeed, if the target node has surely listened once, it has received the beacon correctly or seen a collided packet and turned on its data radio in either case. It is possible that nodes that were not the intended target node decided to wake up due to beacon collisions. If they do not receive any traffic in band f_1 after some time, they time out and return to the monitoring state. Eventually, only the desired target nodes keep their data radio on for the duration of the data transfer. The regular MAC layer handles collisions on the data plane.

As an alternative to the beacon-based approach described above, the initiator node could simply send a wakeup tone. In this case, nodes wake up when they detect the presence of signal energy in their listen interval. This is exactly the same situation as when beacons collide. However, with the tone-based approach, a target node never sends back an acknowledgment. As before, the initiator has to send the tone for a sufficiently long tim, such that the target surely has awoken once. Note that all nodes in the neighborhood of the initiator always wake up in this case. In this paper, we analyze both the beacon-based and the tone-based scheme. We refer to them as ***STEM-B*** (beacon) and ***STEM-T*** (tone), respectively.

4 Theoretical Analysis of STEM

4.1 Setup Latency

Before simulating our protocol, we first develop a theoretical model of the system performance. We define the *setup latency T_S of a link* as the interval from the time the initiator starts contacting the target to the time both nodes have turned on their data radio.

In STEM-B, the average setup latency is approximately given by (1) for the case where there is no beacon collision. In this equation, B_1 and B_2 are the transmit duration of the beacon and acknowledgment packet, respectively. To conserve the flow of this paper, we have moved the derivation of (1) to Appendix A.

$$\bar{T}_S = \frac{T + T_B}{2} + 2 \cdot B_1 + B_2 - T_{Rx}. \tag{1}$$

In Appendix A, we also show that the setup latency in case of a beacon collision is given by (2). This expression, at the same time, specifies the maximum time during which the initiator needs to send beacons and is, in essence, the worst-case latency of STEM-B.

$$T_S = T + T_B + 2 \cdot B_1 + B_2 - T_{Rx}. \tag{2}$$

The setup latency of the tone-based variant, STEM-T, is constant and given by (3), as we derive in Appendix A. In this equation, T_I is the time interval over which channel sensing needs to be performed to achieve a satisfactory low false alarm probability.

$$T_S = T - T_{Rx} + 2 \cdot T_I. \tag{3}$$

4.2 Energy Savings

For typical short-range radios used in sensor networks, the transmit, receive, and idle power are almost identical (see also Table 1). For the data radio, we approximate all of them by just one value: P_1. For the wakeup radio, which is not necessarily of the same type as the data radio, this value is P_2. In our analysis, P_1 and P_2 are chosen equal to the idle power. The sleep power for the data and wakeup radio are denoted as $P_{sleep,1}$ and $P_{sleep,2}$, respectively. To simplify our subsequent expressions, we define two new parameters:

$$\rho = \frac{P_2}{P_1}, \tag{4}$$

$$\phi = \frac{P_{sleep,1} + P_{sleep,2}}{P_1}. \tag{5}$$

Parameter ρ represents the relative power of the data and wakeup radio. We have introduced parameter ϕ for readability reasons, but it also corresponds to the lower bound on the power savings, as we will explain in Section 6.3.

After being awakened using STEM, a node turns on its data radio in band f_1. When the data communication phase is over, the node returns to the low-power sleep state. The average time the radio is on during one such data communication phase is denoted as t_{burst}. If the MAC protocol is such that the node turns off its radio for some time, this sleep time is not included in t_{burst}. Each communication phase requires one transition from the monitoring to the transfer state. The number of such transitions per second is called the wakeup frequency, f_W. The fraction of time the data radio is turned on is thus given by α, defined as (6). This also corresponds to the relative importance of the transfer state. The inverse of the duty cycle of the wakeup radio is called β.

$$\alpha = f_W \cdot t_{burst}, \tag{6}$$

$$\beta = \frac{T}{T_{Rx}}. \tag{7}$$

We evaluate the energy savings of running STEM relative to the situation where there is no wakeup radio and the data radio is never turned off. The relative energy for both STEM-B and STEM-T is approximately given by (8). In this equation, f_S is the average number of times per second the node sets up a link as the initiator or, equivalently, the setup frequency. The derivation of this expression is included in Appendix B.

$$\frac{E}{E_0} = \frac{\rho}{\beta} + \alpha + f_S \cdot T_S \cdot \rho + \phi. \tag{8}$$

Although (8) is valid for both versions, α is likely to be larger in STEM-T than STEM-B since more nodes are awakened when they are not the intended target. In any case, the first two terms are typically dominant. The energy savings are larger when β increases, by extending the period T. This results in larger setup latencies, as can be seen from (1)-(3). The energy savings are also larger when the monitoring state becomes more dominant and when fewer setups are needed. If the wakeup radio can be designed to be lower power than the data radio ($\rho < 1$), the savings also increase. The last term in (8) presents a floor to the energy as the best we can do is to have the two radios sleeping all the time. Since the node has a finite battery capacity, the energy savings directly correspond to the same relative increase in the node's lifetime, which ultimately results in a prolonged lifetime of the sensor network.

5 STEM Performance Evaluation

5.1 Simulation Setup

In this section, we verify our algorithm and theoretical analysis through simulations which were written on the Parsec platform, an event-driven parallel simulation language [12]. We distribute N nodes in a uniformly random fashion over a field of size L x L. Each node has a transmission range R. For a uniformly random deployment, the network connectivity is only a function of the average number of neighbors of a node, denoted by parameter λ:

$$\lambda = \frac{N}{L^2} \cdot \pi R^2. \tag{9}$$

Since traffic communication patterns depend solely on the network connectivity, we only have to consider λ and not N, R, and L separately. This statement was verified through simulations and we therefore can characterize a uniform network density by the single parameter λ.

In principle, data and wakeup radios can be different. This is especially true for STEM-T, where the wakeup radio only needs to be able to send out a tone and detect it,

TABLE 2
Simulation Settings

General settings	
R	20 m
L	79.27 m
R_b	2.4 Kbps

STEM-B	
T_{Rx}	225 ms
T_B	150 ms
L_{beacon}	144 bits
L_{ack}	144 bits

STEM-T	
T_{Rx}	10 ms
T_I	9.5 ms

possibly resulting in a simplified implementation. In our simulations, we have chosen the same TR1000 radio of Table 1 for both data and wakeup such that $\rho = 1$. Table 2 lists the other simulation settings. The area of the sensor network is such that, for $N = 100$, we have $\lambda = 20$. Furthermore, our setup includes a CSMA-type MAC, similar to the DCF (distributed coordination function) of 802.11. The node closest to the top left corner detects an event and sends 20 information packets of 1,040 bits (including all headers) to the data sink with an interpacket spacing of 16 seconds. The node turns its data radio back off if it has not received any traffic for 20 seconds. The time for the data transfer, t_{burst}, is thus about 340 seconds. The data sink is the sensor node located closest to the bottom right corner of the field. We have observed that the average path length is between six and seven hops. All reported results are averaged over 100 simulation runs.

Clearly, the nodes that are on the path consume more energy than the ones that are not. For those on the path, f_W is equal to the inverse of the total simulation time since there is only one communication phase. The value of α is thus the same for STEM-B and STEM-T. In addition, f_S is equal to f_W, except for the final destination, where it is zero. Besides the nodes on the path, there are those that are awakened accidentally in STEM-T or due to collisions in STEM-B. They have the same value of f_W as above, but f_S is equal to zero. Therefore, they consume less energy. Finally, all other nodes always remain in the low-power listen state and have both f_W and f_S equal to zero. In our subsequent simulations, we only report the average energy for the nodes that are on the path. This is essentially the worst case as all other nodes will consume less energy and are therefore less critical in the considered scenario. If we had averaged over all nodes, the energy savings would depend on the size of the overall network. The effect of waking up nodes that are not in the path is addressed in Section 6.

5.2 Simulation Results

Fig. 5 shows the average setup latency per hop as a function of the wakeup period T. The simulation results of STEM-B without collisions agree well with the theoretical analysis of (1). It also confirms that the approximations used in Appendix A are indeed appropriate for the chosen settings. In addition, we have verified that, if the maximum beacon train duration is equal to (2), at least one beacon is received. The worst-case latency for STEM-B in the case of collisions is thus given by (2), which is also plotted in Fig. 5. For STEM-T, the setup latency is equal to the chosen tone duration, given by (3). We have verified that the short listen time is indeed never missed.

For the same average setup latency, the period T of STEM-B can be approximately twice as long as that of STEM-T, in case there are no collisions. The reason is the feedback provided by the acknowledgments in STEM-B. Note that we have used a relatively slow radio with a bit-rate of just 2.4 Kbps. By choosing a radio that is 10 times faster, the absolute latency also decreases by this factor.

Figs. 6 and 7 show the relative energy of STEM-B and STEM-T as a function of the period T for different values of α. As defined in the previous section, α represents the fraction of time in the transfer state. The solid theoretical curves are obtained from (8) and we observe again the close correspondence to simulated values. As α decreases, the monitoring state becomes more predominant. STEM already results in energy savings when the network is in the monitoring state half of the time. For STEM-T, the two top curves flatten out when the energy reaches α. This is indeed the best gain possible in (8) for a certain amount of data traffic.

When comparing Figs. 6 and 7, we see that STEM-T results in substantially more energy savings than STEM-B. The reason is the following: As discussed in Section 3.3, the values of T_{Rx} have to satisfy (10) and (11) for STEM-B and STEM-T, respectively. Since T_I is typically less than both B_1 and B_2, T_{Rx} is at least three times shorter for STEM-T than STEM-B. Typically, as in our settings, T_{Rx} is much shorter for STEM-T. From (7), we therefore notice that β is larger for STEM-T, which makes it superior in terms of energy savings.

STEM-B $$T_{Rx} \geq T_B + B_1 \geq 2 \cdot B_1 + B_2, \tag{10}$$

STEM-T $$T_{Rx} \geq T_I. \tag{11}$$

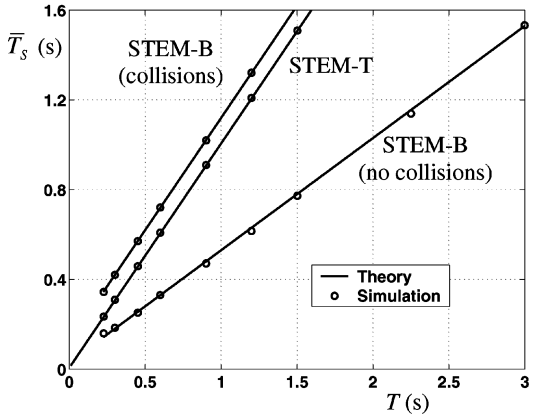


Fig. 5. Average setup latency per hop.

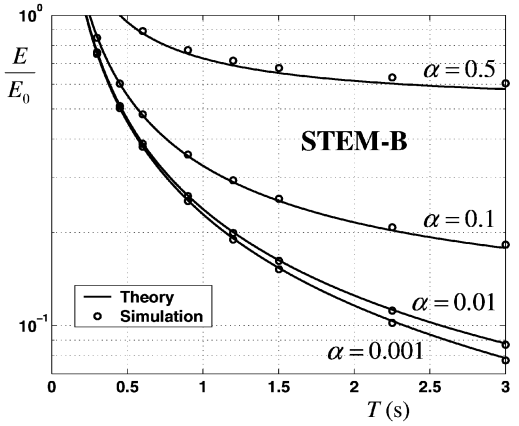


Fig. 6. Energy versus period for STEM-B.

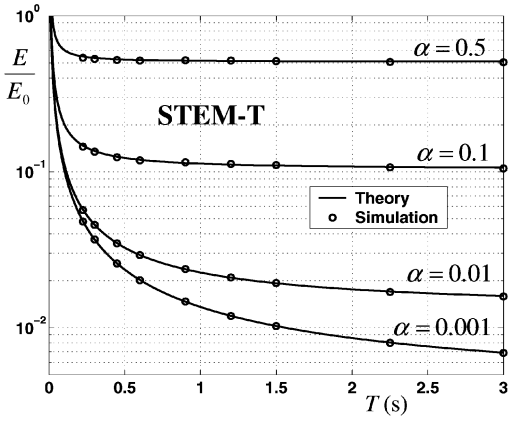


Fig. 7. Energy versus period for STEM-T.

For the same setup latency, STEM-T needs to have a period T that is about twice as large as that of STEM-B, but β is still larger, as argued above. This makes STEM-T preferable in trading energy versus setup latency. However, the disadvantage is that not only the intended node wakes up. This was not an issue in the particular scenario considered here, but can be significant in general. This aspect may eventually negate the edge STEM-T has over STEM-B and make it less efficient. In the next section, where we combine STEM with density-based topology management schemes, this effect is indeed observed to be important.

6 STEM and Network Density

As mentioned in the introduction, existing topology management schemes, such as GAF and SPAN, coordinate the radio sleep and wakeup cycles while ensuring adequate communication capacity. The resulting energy savings increase with the network density. STEM, on the other hand, leverages the setup latency. Moreover, it can be integrated with schemes, such as GAF or SPAN, to achieve additional gains by also exploiting the density dimension in topology management. We specifically focus on combining STEM with GAF.

6.1 Behavior of GAF

In this section, we discuss plain GAF, i.e., without STEM. Furthermore, we also analyze its behavior theoretically as this is an essential building block in the analysis of STEM combined with GAF. Such an analysis was not provided in the original paper [6]. The GAF algorithm is based on a division of the sensor network in a number of virtual grids of size r by r. The value of r is chosen such that all nodes in a grid are equivalent from a routing perspective. In [6], it was derived that r has to satisfy:

$$r \leq \frac{R}{\sqrt{5}}. \tag{12}$$

As before, R denotes the radio transmission range. The average number of nodes in a grid, M, is given by (13). By combining this with (12), we see that M is related to the network density λ by satisfing (14). In the remainder of this paper, we choose (13) and (14) to hold with equality.

$$M = \frac{N}{L^2} \cdot r^2, \tag{13}$$

$$M \leq \frac{\lambda}{5\pi}. \tag{14}$$

Since all nodes in a grid are equivalent from a routing perspective, we can use this redundancy to increase the network lifetime. GAF only keeps one node awake in each grid, while the other nodes turn their radio off. To balance out the energy consumption, the burden of traffic forwarding is rotated between nodes. In the theoretical analysis, we ignore the unavoidable time overlap of this process associated with handoff. If there are m nodes in a grid, the node will (ideally) only turn its radio on $1/m$th of the time and, therefore, last m times longer. The relative energy compared to a scenario without GAF for a node in a grid with m nodes is therefore given by:

$$\left.\frac{E}{E_0}\right|_{node} = \frac{1}{m}. \tag{15}$$

We can essentially view a grid as being a "virtual node," composed of m actual nodes. For the special case of a uniformly random node deployment, the probability of having m nodes in a grid is:[1]

$$Q(m) = \frac{M^m}{m!} \cdot e^{-M}. \tag{16}$$

The derivation of this equation is similar to that of the degree of a node in [4]. However, some grids will not contain any nodes at all and the probability of having m nodes in a used grid is:

$$Q(m|m \geq 1) = \frac{Q(m)}{Q(m \geq 1)} = \frac{M^m}{m!} \cdot \frac{e^{-M}}{1 - e^{-M}}. \tag{17}$$

By combining (15) and (17), we derive that the average relative energy of a node when running GAF is expressed as (18).

$$\left.\frac{E}{E_0}\right|_{node} = \frac{1 - e^{-M}}{M}. \tag{18}$$

1. We use the symbol Q in this paper for probabilities to avoid confusion with power (denoted by P).

6.2 Analysis of STEM combined with GAF

As discussed in the previous subsection, GAF leverages the network density to conserve energy while leaving the data forwarding capacity intact. STEM, on the other hand, saves energy by trading it for path setup latency. We anticipate better results by combining both approaches in an effort to exploit both latency and density dimensions.

In GAF, a grid can be viewed as having one virtual node and the physical nodes alternatively perform the functionality of that virtual node. From this perspective, STEM can be introduced in a straightforward manner by letting it run on the virtual node. In real life, nodes alternate between sleep and active states, as governed by GAF. The one active node in the grid runs STEM in the same way as described in Section 3. The routing protocol only needs to be modified to address virtual nodes (or grids) instead of real nodes.

However, we need to change the mechanism by which the functionality of being active in a grid is rotated between nodes, which is referred to as "leader election." In the original election scheme of GAF, described in [6], nodes that are asleep decide to become the leader after some time interval. To resolve the inconsistency of having multiple leaders, these nodes send periodic broadcasts and listen to similar messages from the other leaders in their grid. Upon receiving such broadcasts, each leader decides to go to sleep or remain a leader based on the expected remaining time to live of both nodes, which is included in the broadcasts. Note that this procedure requires the leader to have its radio on continuously.

If leaders run STEM, as we propose in our hybrid scheme, they have their data radio turned off and will not receive the broadcast messages. We therefore need another election scheme to avoid the persistent occurrence of multiple leaders is one grid. As a solution, a node that wants to become the leader first sets up a link to the current leader using regular STEM. It does not need to know the exact node to address as it can simply wake up "whoever is the current leader." Once the link is set up, the necessary information to decide the election process is exchanged on the data plane. If a node cannot contact the current leader, it assumes that it died (e.g., due to physical destruction) and takes over its role.

With this modification, STEM and GAF can be integrated effectively. As they are orthogonal in our hybrid scheme, we can directly obtain (19) for the relative energy gain of a node in a grid with m nodes. This is based on expanding (8), where the statistics of m are given by (17). The extra term Δ represents the overhead of the leader election process (which we ignored previously in our analysis of GAF). As it is based on STEM, we could model this election overhead using (8), at least in principle. However, quantifying the associated values of wakeup frequency is hard and we chose not to model the overhead in detail.

$$\left.\frac{E}{E_0}\right|_{node} = \frac{1}{m}\left[\frac{\rho}{\beta} + \alpha + f_S \cdot T_S \cdot \rho\right] + \phi + \Delta. \qquad (19)$$

From (19), the average relative energy over all nodes can be derived as being equal to (20), the same way as was done in Section 6.1.

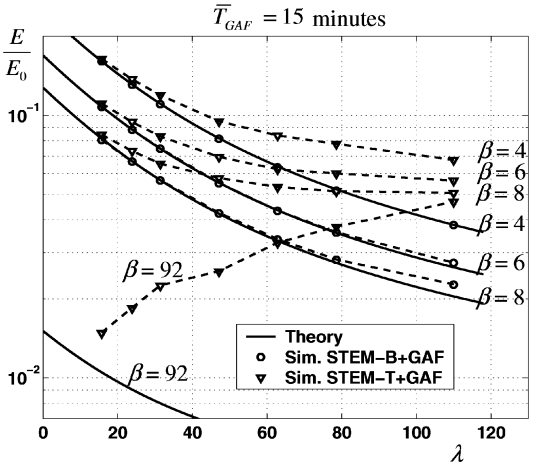


Fig. 8. Relative energy saving versus density for $\overline{T}_{GAF}$ = 15 minutes.

$$\frac{E}{E_0} = \frac{1 - e^{-M}}{M}\left[\frac{\rho}{\beta} + \alpha + f_S \cdot T_S \cdot \rho\right] + \phi + \Delta. \qquad (20)$$

For the link setup latency of regular data traffic, the expressions are exactly the same as the ones for STEM, given in Section 4.1. The reason is that the leader appears simply as a virtual node that is using STEM as long as there is no interference from the leader election process. As this election process occurs at a timescale that is much larger than the link setup time, such interference is negligible.

6.3 Evaluation of STEM Combined with GAF

We now verify our hybrid scheme of STEM combined with GAF through simulations, again with the settings of Tables 1 and 2. To limit the dimensionality of the graphs, we have chosen $\alpha = f_S = 0$. This corresponds to a network that is always in the monitoring state, but we have verified that the algorithm and analysis also work fine when there is data traffic. All reported results are averaged over 1,000 simulation runs. In Fig. 8, the relative energy is plotted versus the network density λ for our hybrid scheme of STEM + GAF. We can also deduce the behavior of pure STEM without GAF from this figure. By comparing (8) and (20), we see that the behavior of STEM alone is mathematically equivalent to that of STEM+GAF with $M = \lambda = 0$ and $\Delta = 0$. Although, $\lambda = 0$ has no physical significance by itself, it allows us to visualize the behavior of STEM on the same graph as that of STEM + GAF.

A node tries to become the leader after a random time T_{GAF} in the range of 15 $\pm$ 3 minutes. For the theoretical values, we have set $\Delta = 0$ due to the complexity of modeling the leader-election overhead. This causes the discrepancies in Fig. 8 between the theoretical analysis and the simulated results, denoted by circles for STEM-B and by triangles for STEM-T. For the same value of the inverse duty cycle β, STEM-B outperforms STEM-T, although both are given by (19). The reason is that, in STEM-T, the leader wakes up each time one of its neighbors initiates the election process, even if it is not part of the same virtual grid. In STEM-B, the leader only wakes up as a response to other nodes in its grid. However, for the same β, the setup latency of STEM-T is smaller than that of STEM-B.

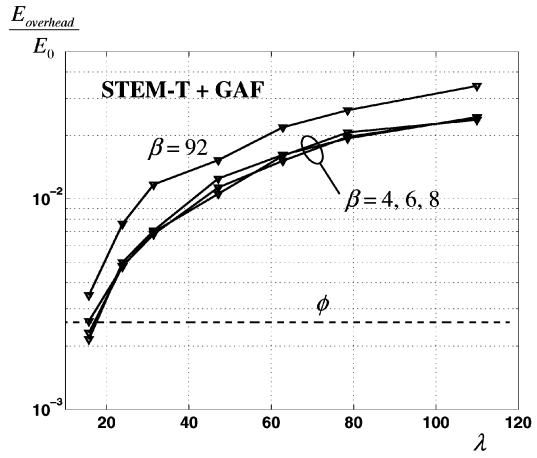


Fig. 9. Leader election overhead for STEM-T.

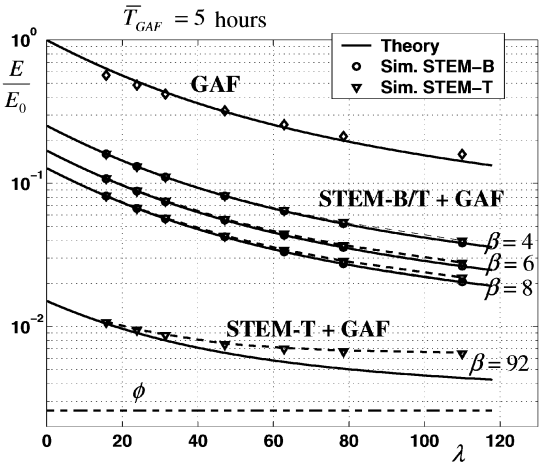


Fig. 10. Relative energy saving versus density for TGAF = 5 hours.

To compare both schemes for similar latency, we have included the curve for STEM-T with $\beta = 92$. This results in a setup latency of 0.93 seconds, which is the same as that of STEM-B with $\beta = 8$. The unexpected behavior of the simulated curve with $\beta = 92$ is due to the GAF leader-election overhead. As we will illustrate shortly, this overhead does not vary much with β and increases with λ. As the energy savings due to STEM are large for large β, the overall energy consumption of STEM+GAF is heavily dominated by the leader-election overhead, in this case, as it becomes the major cause of energy consumption.

For the same setup latency, one of the two variants (STEM-T+GAF with $\beta = 92$ or STEM-B+GAF with $\beta = 8$) is more energy efficient, depending on the network density, with a crossover point at $\lambda = 62$. Note that, for these particular settings, it is undesirable to leverage the density using GAF. Instead, running STEM-T without GAF, thereby avoiding the leader election overhead, is most energy efficient.

Fig. 9 plots the overhead of the leader election for STEM-T. When the network density increases, leaders are awakened more frequently. In addition, the effect of the overhead becomes relatively more pronounced in Fig. 8 when the absolute energy decreases. For STEM-B, the overhead is considerably below the value of ϕ and is not visible in Fig. 9.

In the previous simulations, a node tries to become a leader relatively often (about every 15 minutes). In more realistic scenarios, the election process is likely to operate at a much larger timescale such that the overhead decreases. Indeed, a node is expected to live longer when its energy consumption is reduced (ignoring physical destruction). It can therefore remain the leader for a longer time period. Fig. 10 shows the relative energy savings of the different schemes for T_{GAF} in the range of 5 ± 1 hour. The effect of the overhead is heavily reduced in this case, except when the absolute energy is extremely low. From (19), we also see that the absolute best we can do is have all nodes sleeping all the time such that the relative energy is given by ϕ. Although not shown here, we also verified that the link setup latency is similar to that of STEM alone.

Fig. 10 also shows the results of GAF alone and, as explained before, the behavior of pure STEM corresponds to the point on the curves of STEM + GAF where $\lambda = 0$. By combining STEM and GAF as in our hybrid scheme, both network density and path setup latency are leveraged to achieve considerable energy savings. Even at low densities or low latencies, the other dimension can be traded off for energy savings. The gains are compounded when both dimensions can be exploited together. For a network density of $\lambda = 80$ and a setup latency per hop of 0.93 seconds (STEM-T with $\beta = 92$), the energy consumption of a node is 150 times lower than without topology management. In Fig. 10, we see that this is already close to the lower bound of ϕ.

Our hybrid scheme provides the sensor network designer with full flexibility to trade energy, latency, and density for each other. The equations we derived in the previous section allow him/her to predict, at design-time, the exact relationships between these three parameters when our hybrid scheme is deployed in the network. For example, given a bound on the maximum allowable latency and the desired energy consumption per node, we can calculate the required density and listen period of STEM. With these settings, the network will satisfy the aforementioned constraints when running our hybrid STEM+GAF topology management protocol.

7 Conclusions

In this paper, we have introduced STEM, a topology management technique that trades power savings for path setup latency in sensor networks. It emulates a paging channel by having a separate radio operating at a lower duty cycle. Upon receiving a wakeup message, it turns on the primary radio, which takes care of the regular data transmissions. This wakeup message can take the form of a beacon packet or simply a tone, resulting in two variants of STEM. Our topology management is specifically geared toward those scenarios where the network spends most of its time waiting for events to happen, without forwarding traffic.

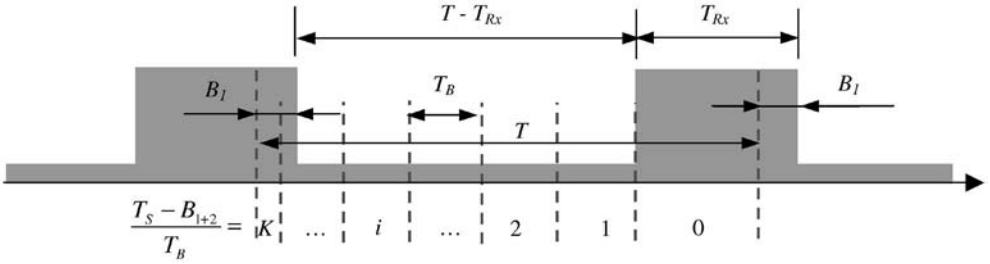


Fig. 11. Analysis of the setup latency of STEM-B without collisions.

Whether the beacon-based or the tone-based variant is superior depends on the application scenario. We observed this when combining STEM with a topology management scheme that leverages the network density. The resulting hybrid scheme exploits both setup latency and network density. However, whether density should be leveraged and which variant of STEM is most energy efficient, depends on the impact of the protocol overhead. For practical settings, the combination of STEM and GAF can reduce the energy to 1 percent or less of that of a network without topology management. Alternatively, this results in an increase of the average node lifetime of a factor 100.

At design time, the settings of our protocol can be derived, positioning the network at the desired operating point in the density-latency-energy design space. In essence, this is just part of a more general and design tradeoff, which is impacted by the specific application, the layout of the network, the cost of the nodes, the desired network lifetime, and many other factors.

Appendix A

Derivation of the Setup Latency

First, we derive the setup latency (as defined in Section 4.1) for STEM-B. If there are no collisions, the target sends back an acknowledgment after it receives a beacon packet. As soon as the initiator receives this acknowledgment, the link is set up. The setup latency is thus a number of beacon periods plus the time to transmit the beacon that is received and get back an acknowledgment. Therefore, T_S is equal to $B_1 + 2$ plus an integer multiple of the interbeacon spacing T_B, where we use the shorthand notation $B_1 + 2 = B_1 + B_2$.

As the target and originator node are not synchronized, the beacon sending process starts at a random point in the cycle T of the target node. Fig. 11 shows the normalized values of T_S for different start times of the beacon sending process. In the region that is labeled i (i = 1..K), the setup latency is equal to $i \cdot T_B + B_1 + 2$. The reason is that beacon $i + 1$ is the first one to fall entirely within the interval of length T_{Rx} when the target node's radio is on. The probability of being in region i is equal to the length of that region divided by T. As a result, for $T > T_{Rx}$, the statistics of T_S are derived from Fig. 11 as:

$$\begin{cases} P(T_S = B_{1+2}) = \frac{T_{Rx} - B_1}{T} \\ P(T_S = k \cdot T_B + B_{1+2}) = \frac{T_B}{T} \qquad k = 1...K \\ P(T_S = (K+1) \cdot T_B + B_{1+2}) = \frac{T - (T_{Rx} - B_1) - K \cdot T_B}{T} \end{cases} \quad \text{(A1)}$$

$$K = \left\lfloor \frac{T - (T_{Rx} - B_1)}{T_B} \right\rfloor.$$

Based on this equation and after some algebra, the average setup latency per hop can be calculated as being equal to:

$$\bar{T}_S = B_{1+2} + \frac{T - T_B}{2} - \varepsilon \cdot \left(1 - \frac{T_B + \varepsilon}{2 \cdot T}\right) + \delta \cdot (1 - \delta) \cdot \frac{T_B^2}{2 \cdot T}. \quad \text{(A2)}$$

The variables δ and ε, which we introduced to simplify the notation of (A2), are defined as:

$$\delta = \frac{T - (T_{Rx} - B_1)}{T_B} - K, \quad \text{(A3)}$$

$$\varepsilon = T_{Rx} - T_B - B_1. \quad \text{(A4)}$$

We have verified that, in practical scenarios, the last term in (A2) is negligible, resulting in:

$$\bar{T}_S = B_{1+2} + \frac{T - T_B}{2} - \varepsilon \cdot \left(1 - \frac{T_B + \varepsilon}{2 \cdot T}\right), \quad \text{(A5)}$$

In addition, T is typically substantially larger than T_{Rx} such that we can further simplify this expression to:

$$\begin{aligned} \bar{T}_S &= B_{1+2} + \frac{T - T_B}{2} - \varepsilon \\ &= \frac{T + T_B}{2} + 2 \cdot B_1 + B_2 - T_{Rx}. \end{aligned} \quad \text{(A6)}$$

From (A1) and Fig. 10, we also learn that the maximum setup latency is equal to:

$$T_S^{\max} = (K+1) \cdot T_B + B_{1+2}. \quad \text{(A7)}$$

Furthermore, K is maximal when (A3) is equal to 0. Therefore, we can bound the setup latency by:

$$T_S^{\max} \leq T - \varepsilon + B_{1+2}. \quad \text{(A8)}$$

If the initiator sends out beacons for this time period, the target is guaranteed to have received it unless there was a collision or the packet got corrupted. However, in the case of a collision or corrupted packet, the receiver has

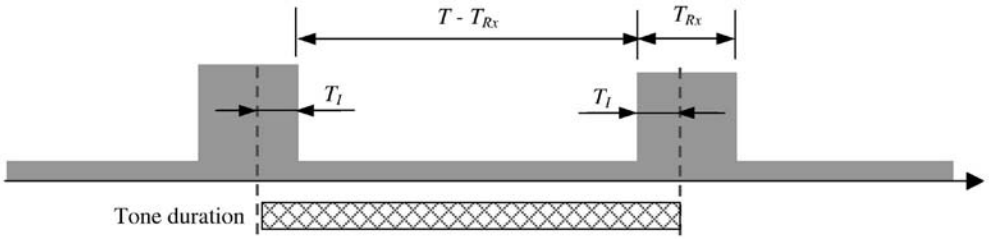


Fig. 12. Analysis of the worst-case setup latency of STEM-T.

awakened as well, as we explained in Section 3.3. In any case, the initiator can stop the beacon sending process after T_S^{max}, given by (A8). When a beacon collision occurs, the setup latency is also given by this equation.

The analysis of STEM-T is similar to the collision scenario in STEM-B. For sufficient noise tolerance, we propose an integrative detector. The target detects the tone if it is integrated for at least an interval T_I. Fig. 12 shows the worst-case scenario, where the tone starts too late to be detected in the first wakeup period. From this figure, we see that the minimum tone duration to guarantee detection is equal to (A9), which is therefore also the setup latency.

$$T_S = T - T_{Rx} + 2 \cdot T_I. \tag{A9}$$

APPENDIX B

DERIVATION OF THE ENERGY CONSUMPTION

When running STEM, the total energy consumed by a node during a time interval t can be broken up into two components, one for each frequency band.

$$E_{node} = E_{wakeup} + E_{data}. \tag{A10}$$

Equation (A11) details the energy consumption in the wakeup plane. The first term accounts for the listening cycle, where $P_{node,2}$ is given by (A12). $P_{listen,2}$ denotes the power during the periodic listen interval, which contains contributions of idle and receive power. The second term in (A11) represents the energy of sending beacon packets and listening for the acknowledgment (STEM-B) or the energy of sending a tone (STEM-T). It therefore corresponds to the energy spend as an initiator, where the average power $P_{setup,2}$ has contributions of transmission, reception, and idle power. $P_{sleep,2}$ is the sleep power of the wakeup radio.

$$E_{wakeup} = P_{node,2} \cdot (t - t_{setup}) + P_{setup,2} \cdot t_{setup}, \tag{A11}$$

$$P_{node,2} = \frac{P_{sleep,2} \cdot (T - T_{Rx}) + P_{listen,2} \cdot T_{Rx}}{T}. \tag{A12}$$

The energy consumption in the data plane is given by (A13). In this equation, t_{data} is the total time the radio is turned on in the data plane. As a result, $P_{data,1}$ contains contributions of packet transmission, packet reception, and idle power.

$$E_{data} = P_{sleep,1} \cdot (t - t_{data}) + P_{data,1} \cdot t_{data}. \tag{A13}$$

For ease of comparison, we normalize the energy consumption of STEM to a scenario where there is only one radio that is never in the sleep state, see (A14) and (A15). In (A15), P_1 is as defined in Section 4.2.

$$\frac{E}{E_0} = \frac{E_{node}}{E_{node}^{original}}, \tag{A14}$$

$$E_{node}^{original} = P_1 \cdot t. \tag{A15}$$

As explained in Section 4.2, we approximate $P_{data,1} \approx P_1$ and $P_{setup,2} \approx P_{listen,2} \approx P_2$. Furthermore, we note that $P_{sleep,i} << P_i$ $(i = 1, 2)$, which allows us to write the relative energy of (A14) as (A16), after appropriate simplifications. Variables ρ and ϕ are defined in (4) and (5).

$$\frac{E}{E_0} = \frac{T_{Rx}}{T} \cdot \rho + \frac{t_{data}}{t} + \frac{t_{setup}}{t} \cdot \left(1 - \frac{T_{Rx}}{T}\right) \cdot \rho + \phi. \tag{A16}$$

In this equation, t_{setup} is the total time spent setting up the link in the wakeup plane as an initiator. The ratio of t_{setup} and the total time t can be rewritten as the setup frequency, f_S, times the duration of one setup T_S. When T is not too small, we can also make the following simplification:

$$\left(1 - \frac{T_{Rx}}{T}\right) \approx 1. \tag{A17}$$

Similarly, t_{data} can be split up in bursts of average duration t_{burst}, where a burst of data transfer requires one link setup. Consequently, the fraction of time the data-plane radio is turned on, which we define as α, is written as (A18), corresponding to (6). Here, f_W is the wakeup frequency. It can be higher than the setup frequency as a node wakes up when it is the initiator, the target, or an unintended receiver that hears a collision (STEM-B) or a tone (STEM-T).

$$\alpha = \frac{t_{data}}{t} = f_W \cdot t_{burst}. \tag{A18}$$

Finally, by defining β as in (7), (A16) becomes:

$$\frac{E}{E_0} = \frac{\rho}{\beta} + \alpha + f_S \cdot T_S \cdot \rho + \phi. \tag{A19}$$

If the wakeup radio would be turned off once the data radio is turned on, the derivation is similar to the one above. With the same simplifications, the final equation is exactly the same as (A19). This situation corresponds to STEM-T, where there is no need to leave the wakeup radio on once the data radio has been activated.

ACKNOWLEDGMENTS

This paper is based in part on research funded by the US Office of Naval Research and the US Defense Advanced Research Projects Agency PAC/C and SenseIT programs through US Air Force Research Laboratory contracts F30602-

00-C-0154 and F30602-99-1-0529. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ONR, DARPA, Air Force Rome Laboratory, or the US Government.

REFERENCES

[1] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Comm. Magazine,* vol. 7, no. 5, pp. 16-27, Oct. 2000.

[2] L. Clare, G. Pottie, and J. Agre, "Self-Organizing Distributed Sensor Networks," *SPIE—The Int'l Soc. Optical Eng.,* pp. 229-237, Apr. 1999.

[3] D. Estrin and R. Govindan, "Next Century Challenges: Scalable Coordination in Sensor Networks," *Proc. MobiCom 1999,* pp. 263-270, Aug. 1999.

[4] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy-Aware Wireless Sensor Networks," *IEEE Signal Processing,* vol. 19, no. 2, pp. 40-50, Mar. 2002.

[5] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *Proc. MobiCom 2001,* pp. 70-84, July 2001.

[6] Y. Xu, J. Heidemann, and D. Estrin, "Geography-Informed Energy Conservation for Ad Hoc Routing," *Proc. MobiCom 2001,* pp. 70-84, July 2001.

[7] J.-H. Chang and L. Tassiulas, "Energy Conserving Routing in Wireless Ad-Hoc Networks," *Proc. INFOCOM 2000,* pp. 22-31, Mar. 2000.

[8] J. Rabaey, J. Ammer, J.L. da Silva, and D. Patel, "PicoRadio: Ad-Hoc Wireless Networking of Ubiquitous Low-Energy Sensor/Monitor Nodes," *Proc. IEEE CS Workshop VLSI 2000,* pp. 9-12, Apr. 2000.

[9] W. Rabiner Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proc. Hawaii Int'l Conf. System Sciences 2000,* Jan. 2000.

[10] C. Guo, L. Zhong, and J. Rabaey, "Low-Power Distributed MAC for Ad Hoc Sensor Radio Networks," *Proc. Internet Performance Symp. (Globecom '01),* Nov. 2001.

[11] Sensoria Corporation, http://www.sensoria.com/, Year?

[12] R. Bagrodia, R. Meyer, M. Takai, Y.A. Chan, X. Zeng, J. Marting, and H.Y. Song, "Parsec: A Parallel Simulation Environment for Complex Systems," *Computer,* vol. 31, no. 10, pp. 77-85, Oct. 1998.

[13] M. Yacoub, *Foundations of Mobile Radio Engineering.* CRC Press, 1993.

[14] M. McGlynn and S. Borbash, "Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks," *Proc. MobiHoc 2001,* pp. 137-145, Oct. 2001.

[15] "ASH Transceiver Designer's Guide," http://www.rfm.com, Year?

[16] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *IEEE Infocom '02,* June 2002.

[17] "MICA motes at Crossbow," http://www.xbow.com/Products/Wireless_Sensor_Networks.htm, Year?

[18] J. Proakis, *Digital Communications,* third ed. McGraw-Hill Series in Electrical and Computer Engineering, 1995.

Curt Schurgers received the MSEE degree, summa cum laude, from the Katholieke Universiteit Leuven (KUL), Belgium, in 1997. From 1997 to 1999, he worked at IMEC (Interuniversity Micro Electronics Center, Leuven, Belgium) on memory optimization techniques for turbo codes. Currently, he is pursuing a PhD degree at the University of Califirnia at Los Angeles, focusing on energy efficient communication and networking systems. He has received the F.W.O., the B.A.E.F., and the UCLA fellowships in 1997, 1999, and 2000, respectively. He is a student member of the IEEE.

Vlasios Tsiatsis received the BS degree from the Technical University of Crete, Chania, Greece, in 1998, and the MS degree in electrical engineering from the University of California, Los Angeles (UCLA), in 2001. He is currently pursuing a PhD degree at UCLA, researching low-power protocol architectures. In 1998 and 2000, he was honored with the UCLA fellowship.

Saurabh Ganeriwal received the Btech and Mtech degrees in the field of communication and signal processing from the Indian Institute of Technology, Bombay, India, in 2001. Since 2001, he has been pursuing an MS degree at the University of California, Los Angeles. His research interests are in wireless communications and networking. Currently, his research focuses on sensor networks.

Mani Srivastava received the BTech degree in electrical engineering from IIT Kanpur in India and the MS and PhD degrees from the University of California at Berkeley. He is an associate professor of electrical engineering at the University California at Los Angeles. From 1992 through 1996, he was a member of the technical staff at Bell Laboratories in networked computing research. His current research interests are in mobile and wireless networked computing systems, low-power systems, and sensor networks. He received the US National Science Foundation CAREER award in 1997 and the President of India Gold Medal in 1985. He is a member of the IEEE.

Appendix J:

Topology Management for Sensor Networks: Exploiting Latency and Density

Curt Schurgers Vlasios Tsiatsis Saurabh Ganeriwal Mani Srivastava ‡

Networked and Embedded Systems Lab (NESL), Electrical Engineering Department, UCLA
56-125B Eng. IV, UCLA -EE Dept., Los Angeles, CA 90095
{curts, tsiatsis, saurabh, mbs}@ee.ucla.edu

‡ *Please send all correspondence to Mani Srivastava*

ABSTRACT

In wireless sensor networks, energy efficiency is crucial to achieve satisfactory network lifetime. In order to reduce the energy consumption of a node significantly, its radio needs to be turned off. Yet, some nodes have to participate in multi-hop packet forwarding. We tackle this issue by exploiting two degrees of freedom in topology management: the path setup latency and the network density. First, we propose a new technique called Sparse Topology and Energy Management (STEM), which aggressively puts nodes to sleep. It provides a method to wake up nodes only when they need to forward data, where latency is traded off for energy savings. Second, STEM integrates efficiently with existing approaches that leverage the fact that nearby nodes can be equivalent for traffic forwarding. In this case, an increased network density results in more energy savings. We analyze a hybrid scheme, which takes advantage of both setup latency and network density to increase the nodes' lifetime. Our results show improvements of nearly two orders of magnitude compared to sensor networks without topology management.

Keywords : Sensor networks, energy efficiency, topology management.

1. INTRODUCTION

1.1. Sensor Networks

Advances in microelectronic fabrication have allowed the integration of sensing, processing and wireless communication capabilities into low-cost and small form-factor embedded systems called sensor nodes [1][2]. The need for unobtrusive and remote monitoring is the main motivation for deploying a sensing and communication network (sensor network) consisting of a large number of these battery-powered nodes. For example, such systems could be used either outdoors in inhospitable habitats, disaster areas, or indoors for intrusion detection or equipment monitoring. The nodes gather various sensor readings, process them and forward the processed information to a user or, in general a data sink. This forwarding typically occurs via other nodes using a flat or clustered multi-hop path [3][9]. Thus a node in the network essentially performs two different tasks: (1) sensing its environment and processing the information and, (2) forwarding traffic as an intermediate relay in the multi-hop path.

However, the convenience of autonomous remote monitoring comes at a price: an extreme design focus must be placed on energy efficiency as the sensor nodes operate on a small battery with limited capacity [1][2][3]. It is important to view the problem as one of extending the lifetime of the network, rather than just that of the individual nodes. Thus, in addition to improving the efficiency of the nodes, techniques that tackle the problem on the level of the entire network are necessary. This is especially true for the traffic forwarding functionality of the network, as the main energy consumer in a node is the communication subsystem [1][3][4]. Our paper explores this category of network-wide techniques, more specifically dealing with topology management.

1.2. Topology Management

Topology management is an important issue because the only way to save power consumption in the communication subsystem is to completely turn of the node's radio, as the idle mode is almost as power hungry as the transmit mode [4]. However, as soon as a node powers down its radio, it is essentially disconnected from the rest of the network topology and therefore can no longer perform packet relaying. For simplicity, we refer to this state as the node being asleep, although only its radio is turned off. The sensors and processor can still be active, as they are much less power hungry.

The goal of topology management is to coordinate the sleep transitions of all the nodes, while ensuring that data can be forwarded efficiently to the data sink. Existing topology management schemes, such as the ones described in references [5] and [6], are based on the observation that in typical scenarios, some nodes can be asleep without sacrificing significant data forwarding capacity. As density increases, more nodes can be sleeping, resulting in further energy savings. However, major savings would require extremely dense networks, as we will illustrate in this paper.

We propose a different approach to topology management, which exploits the time dimension rather than the density dimension. Strictly speaking, nodes only need to be awake when there is data to forward. We refer to this situation as the network being in the ***'transfer state'***, and in many practical scenarios, this is a rather infrequent event. Most of the time, the sensor network is only monitoring its environment, waiting for an event to happen, and nodes can be asleep. For a large subset of sensor net applications, no data needs to be forwarded to the data sink in this ***'monitoring state'***. Consider for example a sensor network that is designed to detect brush fires. It has to remain operational for months or years, while only sensing if a fire has started. Once a fire is detected, this information should be forwarded to the user quickly. Even when we want to track how the fire spreads, it probably suffices for the network to remain up only for an additional week or so. Similar observations hold for applications such as surveillance of battlefields, machine failures, room occupancy, or other reactive scenarios, where the user needs to be informed once a condition is satisfied.

In the monitoring state, no communication capacity is needed, in principle at least. As there is no data to forward, the communication energy could be completely eliminated, by simply turning off the radios of all nodes. If the need for data forwarding is very rare, the energy savings could be phenomenal. However, there is a crucial caveat: if a node detects an event, it cannot forward the data to the user since all the nodes on the multi-hop path are asleep. If a node has turned off its radio, it will stay completely oblivious of the efforts of other nodes to communicate with it. This is the main dilemma in topology management for sensor nets: a node's radio should be turned off to save energy, yet be left on so the node can know when other nodes need it to forward their traffic. Our topology management scheme, called ***STEM (Sparse Topology and Energy Management)***, solves this issue and trades off energy consumption versus latency of switching back to the transfer state.

Furthermore, we would like to develop a topology management scheme that marries the benefits of both classes discussed previously, namely those that exploit network density and those that exploit setup latency. Ideally, this hybrid solution combines the savings in both dimensions fully, such that a ten-fold energy reduction in both schemes separately would result in a combined hundred-fold reduction. This basically requires these base schemes to be orthogonal in using the independent dimensions of latency and density. We propose such a very effective hybrid scheme in this paper, by combining STEM with techniques that leverage the network density.

2. RELATED WORK

For routing in sensor networks, two alternative approaches have been considered: flat multi-hop and clustering. Although STEM is applicable to both of them, we mainly focus on flat multi-hop routing [3]0[8]. For clustered approaches [9], which are possibly hierarchical, our scheme can be used to reduce the energy of the cluster heads, although the gains are expected to be less dramatic here.

Recently, topology management techniques, called SPAN [5] and GAF [6], have been proposed for flat multi-hop routing. They operate on the assumption that the network capacity needs to be preserved. As a result, the energy consumption is approximately the same whether the network is in the transfer or monitoring state, as no distinction is made between them. Both techniques trade off network density for energy savings. The performance of STEM is independent of network density. It operates in an orthogonal dimension, that of setup latency. Our hybrid scheme, which we describe in section 6, leverages both network density and latency.

With SPAN [5], a limited set of nodes forms a multi-hop forwarding backbone that tries to preserve the original capacity of the underlying ad-hoc network. Other nodes transition to sleep states more frequently, as they no longer carry the burden of forwarding data of other nodes. To balance out energy consumption, the backbone functionality is rotated between nodes, and as such, there is a strong interaction with the routing layer.

Geographic Adaptive Fidelity (GAF) [6] exploits the fact that nearby nodes can perfectly and transparently replace each other in the routing topology. The sensor network is subdivided into small grids, such that nodes in the same grid are equivalent from a routing perspective. At each point in time, only one node in each grid is active, while the others are in the energy-saving sleep mode. Substantial energy gains are, however, only achieved in very dense networks. We will discuss this

issue further on in this paper, when we integrate STEM with GAF.

An approach that is closely related to STEM is the use of a separate paging channel to wake up nodes that have turned off their main radio [10]. However, the paging channel radio cannot be put in the sleep mode for obvious reasons. This approach thus critically assumes that the paging radio is much lower power than the one used for regular data communications. It is yet unclear if such radio can be designed. STEM basically emulates the behavior of a paging channel, by having a radio with a low duty cycle radio, instead of a radio with low power consumption.

The work of McGlynn *et al* [14] describes an algorithm that resembles STEM. However, it is designed to discover the neighbors of all the nodes some time after the network deployment. The goal is to let the network be dormant during deployment, and once the discovery phase starts, learn the complete topology with a high probability. In principle, this algorithm could also be used to set up a path like STEM. However, it is less aggressive, and would result in much larger setup latency, as a node only sends out setup request probabilistically. Furthermore, it does not guarantee discovery of a link.

3. SPARSE TOPOLOGY MANAGEMENT

3.1. Basic Concept

In the application scenarios we consider in this paper, the sensor network is in the monitoring state the vast majority of its lifetime. Ideally, we would like to only turn on the sensors and some preprocessing circuitry. When a possible event is detected, the main processor is woken up to analyze the data in more detail. The radio, which is normally turned off, is only woken up if the processor decides that the information needs to be forwarded to the data sink. Of course, different parts of the network could be in monitoring or transfer state, so, strictly speaking, the 'state' is more a property of the locality of node, rather than the entire network.

Now, the problem is that the radio of the next hop in the path to the data sink is still turned off, if it did not detect that same event. As a solution, each node periodically turns on its radio for a short time to listen if someone wants to communicate with it. The node that wants to communicate, the ***'initiator node'***, sends out beacons with the ID of the node it is trying to wake up, called the ***'target node'***. In fact, this can be viewed as the initiator node attempting to activate the link between itself and the target node. As soon as the target node receives this beacon, it responds to the initiator node and both keep their radio on at this point. If the packet needs to be relayed further, the target node will become the initiator node for the next hop and the process is repeated.

3.2. Dual Frequency Setup

Once both nodes that make up a link have their radio on, the link is active, and can be used for subsequent packets. In order for actual data transmissions not to interfere with the wakeup protocol, we propose to send them in different frequency bands using a separate radio in each band. Sensor nodes developed by Sensoria Corporation [11], for example, are already equipped with two radios. We will discuss the benefits of this dual radio setup in more detail in the next subsection.

Figure 1 shows the proposed radio setup. The wakeup messages, which were discussed in subsection 3.1, are transmitted by the radio operating in frequency band f_1. We refer to these communications as occurring in the ***'wakeup plane'***. Once the initiator node has successfully notified the target node, both nodes turn on their radio that operates in frequency band f_2. The actual data packets are transmitted in this band, or what we call the ***'data plane'***.

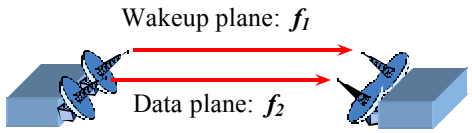


Figure 1 - Radio setup of a sensor node

3.3. STEM Operation

Figure 2 presents an example of typical radio mode transitions for one particular node in the network. Some representative power numbers for the different radio modes are summarized in Table I. These numbers correspond to the TR1000 radio from RF Monolithics [15] where the transmit range is set to approximately 20 meters [4]. This low-power radio has a data rate of 2.4 Kbps and uses OOK modulation.

Table I. Radio power characterization

Radio mode	Power consumption (mW)
Transmit (T_x)	14.88
Receive (R_x)	12.50
Idle	12.36
Sleep	0.016

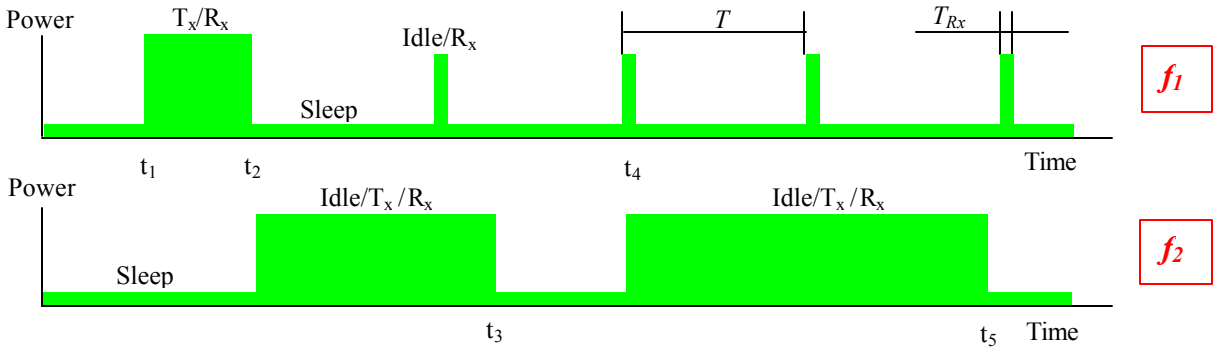


Figure 2 – State transitions of STEM for a particular node

At time t_1, the node wants to wake up one of its neighbors and thus becomes an initiator. It starts sending beacon packets on frequency f_1, until it receives a response from the target node, which happens at time t_2. At this moment, the radio in frequency band f_2 is turned on for regular data transmissions. Note that at the same time, the radio in band f_1 still wakes up periodically from its sleep state to listen if any nodes want to contact it. After the data transmissions have ended (e.g. at the end of a predetermined stream of packets, after a timeout, etc.), the node turns its radio in band f_2 off again. At time t_4, it receives a beacon from another initiator node while listening in the f_1 band. The node responds to the initiator and turns its radio on again in band f_2.

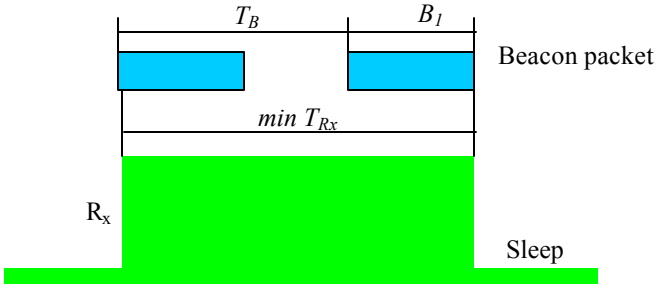


Figure 3 – Radio on-time in the wakeup plane

In order for the target node to receive at least one beacon, it needs to turn on its radio for a sufficiently long time, denoted as T_{Rx}. Figure 3 illustrates the worst-case situation where the radio is turned on just too late to receive the first beacon. In order to receive the second beacon, T_{Rx} should be at least as long as the transmit time B_1 of a beacon packet, plus the inter-beacon interval T_B.

If we were to use one radio operating in just one frequency band, there would be interference between the wakeup and data plane. Consider Figure 4, which shows an ongoing data transfer from node A to B. Node C tries to set up the link to D, and might not be aware of the ongoing transmission. During this polling mode, it aggressively sends beacons in order to avoid missing the short time D is listening. This way, C will use all the channel capacity, and essentially acts as a jammer to B. Despite possible recovery action from the Medium Access Control (MAC) layer, the data communication between A and B will suffer from extra delays. We might allow the setup procedure to be relatively long, as it only occurs once at the start of a communication epoch. However, such long disruptions of ongoing transmissions are typically undesirable. Using one radio that switches between two frequencies could solve this problem, but in that case the regular data transmissions need to be interrupted periodically to listen in the wakeup plane. This is cumbersome, and as integrated radios are ever getting cheaper, we have opted for the dual radio setup. All the results in this paper, however, remain valid for a single radio that switches frequency, but regular data communications will be more complex.

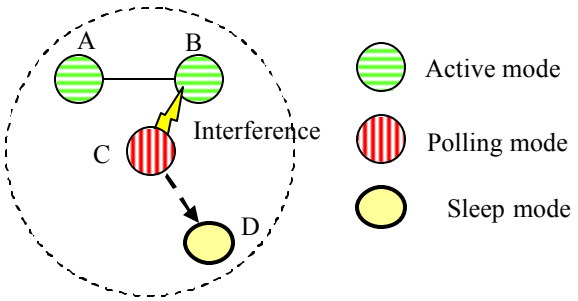


Figure 4 – Interference between the wakeup and the transfer plane in the case of one frequency

Even in the case of two radios, collisions in the wakeup plane are possible. For example, consider Figure 5 that shows a scenario where nodes A and B simultaneously try to wake up the same target node C. In this case the beacons from A and B will collide at C.

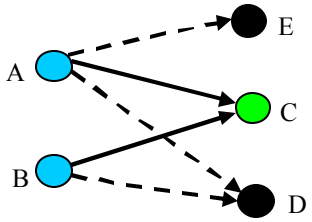


Figure 5 – Collisions on the wakeup plane

To handle this problem, we add extra provisions to the basic STEM operation we discussed thus far. A node also turns on its data radio when there is a collision in the wakeup plane. It does not truly receive packet, but it can detect the presence of signal energy, which is similar to the principle of carrier sensing. In this case, it does not send back an acknowledgement, as it would likely collide with that of other nodes that are also woken up this way. In our example, both C and D turn on their radio in the data plane, since the beacons from A and B collide. Node E receives the beacon from A correctly, and does not wake up, as the beacon tells it that the intended node is C.

After waiting for a response from the target node for time T, the initiator starts transmitting on the data plane. Indeed, the target node will either have received the beacon correctly or seen a collided packet, as it surely has woken up once during this period (see Figure 2). In any case, it has turned on the radio in the data plane. If there is no collision, we chose to send back an acknowledgement, since the initiator knows immediately when the target node is up. This shortens the setup latency, as will also follow from the analytical analysis of section 4.1.

If nodes do not receive data for some time, they time out and go back to sleep. This happens to nodes that were woken up accidentally, like D. Eventually only the desired target node keeps its data-plane radio on for the duration of the data transfer. The regular MAC layer handles any collision that takes place on the data plane.

4. THEORETICAL ANALYSIS OF STEM

4.1. Setup Latency

Before simulating our protocol, we first develop a theoretical model of the system performance. We define the **setup latency T_S of a link** as the interval from the time the initiator starts sending out beacons, to the time both nodes have turned on the radio in the data plane. Typically the target and originator node are not synchronized, which means that the beacon sending process starts at a random point in the cycle of the target node. As a result, the start of the first beacon is distributed uniformly random in interval T. Figure 6 shows the values of T_S, normalized versus the inter-beacon spacing T_B, for different start times of the beacon sending process. Furthermore, the transmission time of a beacon acknowledgment is B_2 and we use the shorthand notation $B_{1+2} = B_1 + B_2$.

First, we carry out this analysis for the case where no collisions take place in the wakeup plane. It is clear that T_S is equal to B_{1+2} plus an integer multiple of T_B. If the initiator node starts the wakeup process in the region that is labeled i in Figure 6 (i = 1..K), the setup latency is equal to $i \cdot T_B + B_{1+2}$. The reason is that beacon $i+1$ is the first one to fall entirely within the interval of length T_{Rx} when the target node's radio is on. The probability of being in region i is equal to the length of that region divided by T. As a result, for $T > T_{Rx}$, the statistics of T_S are derived from Figure 6 as:

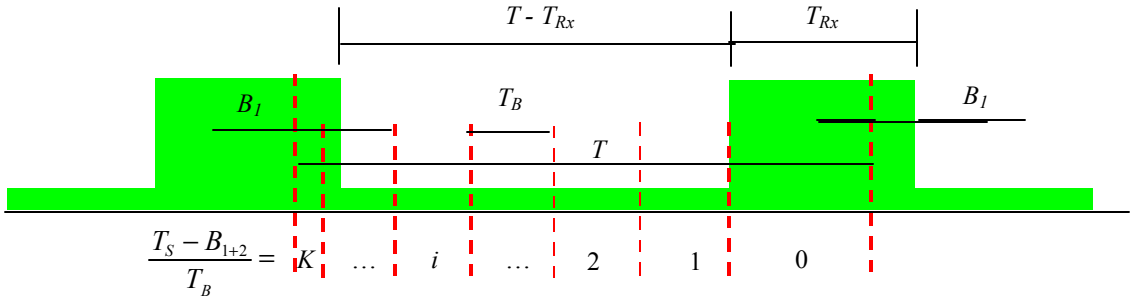


Figure 6 – Analysis of the setup latency

$$\begin{cases} P(T_S = B_{1+2}) = \dfrac{T_{Rx} - B_1}{T} \\ P(T_S = k \cdot T_B + B_{1+2}) = \dfrac{T_B}{T} \qquad k = 1...K \\ P(T_S = (K+1) \cdot T_B + B_{1+2}) = \dfrac{T - (T_{Rx} - B_1) - K \cdot T_B}{T} \end{cases} \quad (1)$$

$$K = \left\lfloor \frac{T - (T_{Rx} - B_1)}{T_B} \right\rfloor$$

Based on this equation, the average setup latency per hop can be calculated as being equal to:

$$\overline{T}_S = B_{1+2} + \frac{T - T_B}{2} - e \cdot \left(1 - \frac{T_B + e}{2 \cdot T}\right) + d \cdot (1-d) \cdot \frac{T_B^2}{2 \cdot T} \quad (2)$$

The variables d and e, which we introduced to simplify the notation of (2), are defined as:

$$d = \frac{T - (T_{Rx} - B_1)}{T_B} - K \quad (3)$$

$$e = T_{Rx} - T_B - B_1 \quad (4)$$

We have verified that in practical scenarios, the last term in (2) is negligible, resulting in:

$$\overline{T}_S = B_{1+2} + \frac{T - T_B}{2} - e \cdot \left(1 - \frac{T_B + e}{2 \cdot T}\right) \quad (5)$$

In addition, T is typically substantially larger than T_{Rx}, such that we can further simplify this expression to:

$$\overline{T}_S = B_{1+2} + \frac{T - T_B}{2} - e$$

$$= \frac{T + T_B}{2} + 2 \cdot B_1 + B_2 - T_{Rx} \quad (6)$$

The above equations are valid on condition that $T > T_{Rx}$. For the special case when there is no sleep period, $T = T_{Rx}$, the average setup delay is equal to:

$$\overline{T}_S = B_{1+2} \quad (7)$$

Thus far, we assumed that there are no collisions in the wakeup plane. If setup packets collide in the wakeup plane, the initiator nodes will eventually time out after time T, as discussed in the previous section. This means that the setup latency in this case is equal to:

$$\overline{T}_S = T \quad (8)$$

4.2. Energy Savings

Next, we derive expressions for the energy savings resulting from running STEM. The total energy consumed by a node during a time interval t can be broken up into two components, one for each frequency band.

$$E_{node} = E_{wakeup} + E_{data} \quad (9)$$

Equation (10) details the energy consumption in the wakeup plane. The first term accounts for the listening cycle, where P_{node} is given by (11). In this equation P^0_{node} is a combination of idle and receive power. The second term in (10) represents the energy of transmitting and receiving beacon and response packets (P_{setup} is thus a combination of transmit, receive and idle power).

$$E_{wakeup} = P_{node} \cdot (t - t_{setup}) + P_{setup} \cdot t_{setup} \quad (10)$$

$$P_{node} = \frac{P_{sleep} \cdot (T - T_{Rx}) + P^0_{node} \cdot T_{Rx}}{T} \quad (11)$$

The energy consumption in the data plane is given by (12). In this equation, t_{data} is the total time the radio is turned on in the data plane. As a result, P_{data} contains contributions of packet transmission, packet reception and idle power.

$$E_{data} = P_{sleep} \cdot (t - t_{data}) + P_{data} \cdot t_{data} \quad (12)$$

Without topology management, the total energy would be equal to (13). Although P_{data} also contains contributions of P_{idle}, we have chosen to split up the energy consumption in analogy with (12) for ease of comparison. The main difference is that the radio is never in the energy-efficient sleep state here.

$$E^{original}_{node} = P_{idle} \cdot (t - t_{data}) + P_{data} \cdot t_{data} \quad (13)$$

We evaluate the benefits of STEM, by considering the relative energy, which is defined as:

$$\frac{E}{E_0} = \frac{E_{node}}{E^{original}_{node}} \quad (14)$$

The energy savings can be evaluated by combining (9)-(14). Since transmit, receive and idle power are very similar, see Table 1, we can approximate $P_{idle} \approx P_{data} \approx P_{setup} \approx P^0_{node} \approx P$. Furthermore, we note that $P_{sleep} \ll P$, which allows us to write the relative energy as (15), after appropriate simplifications.

$$\frac{E}{E_0} = \frac{T_{Rx}}{T} + \frac{t_{data}}{t} + \frac{t_{setup}}{t} \cdot \left(1 - \frac{T_{Rx}}{T}\right) + 2 \cdot \frac{P_{sleep}}{P} \qquad (15)$$

t_{setup} is the total time spent setting up the link in the wakeup plane. We define the time to do one setup as t_{1setup} and the number of such setups per second, or the setup frequency, as f_S. When T is not too small, t_{1setup} is close to $T/2$ if there are no collisions, see (6). In case, we make the following simplifications:

$$\frac{t_{setup}}{t} = t_{1setup} \cdot f_S \approx \frac{T}{2} \cdot f_S \qquad (16)$$

$$\left(1 - \frac{T_{Rx}}{T}\right) \approx 1 \qquad (17)$$

Similarly, t_{data} can be split up in bursts of average duration t_{burst}, where a burst of data transfer requires one link setup. Consequently, the fraction of time the data-plane radio is turned on, which we define as a, can be written as (18). We note that a corresponds directly to the relative importance of the transfer state.

$$a = \frac{t_{data}}{t} = f_S \cdot t_{burst} \qquad (18)$$

Finally, we call b the inverse of the duty cycle in the wakeup plane:

$$b = \frac{T}{T_{Rx}} \qquad (19)$$

With the above definitions and simplifications, (15) can be rewritten as (20) or (21).

$$\frac{E}{E_0} = \frac{1}{b} + a + \frac{f_S \cdot T}{2} + 2 \cdot \frac{P_{sleep}}{P} \qquad (20)$$

$$\frac{E}{E_0} = \frac{1}{b} + f_S \cdot \left(t_{burst} + \frac{T}{2}\right) + 2 \cdot \frac{P_{sleep}}{P} \qquad (21)$$

It is clear that the energy savings are larger when b increases, by extending the period T. This results in larger setup latencies, as can be seen from (6). The energy savings are also larger, when the transfer state becomes smaller, and fewer setups are needed. The last term in (20) and (21) presents a floor to the energy, as the best we can do is to have the two radios sleeping all the time.

Since the node has a finite battery capacity, the energy savings directly correspond to the same relative increase in the node's lifetime, which ultimately results in a prolonged lifetime of the sensor network.

5. STEM PERFORMANCE EVALUATION

5.1. Simulation Setup

In this section, we verify our algorithm and theoretical analysis through simulations, which were written on the Parsec platform, an event-driven parallel simulation language [12]. We distribute N nodes in a uniformly random fashion over a field of size L x L. Each node has a transmission range R.

For a uniform network density, the probability $Q(n)$ for a node to have n neighbors in a network of N nodes is given by the binomial distribution of (22), when edge effects are ignored. In this equation, Q_R is the probability of a node being in the transmission range of a particular node, given by (23).†

$$Q(n) = Q_R^n \cdot (1 - Q_R)^{N-1-n} \cdot \binom{N-1}{n} \qquad (22)$$

$$Q_R = \frac{pR^2}{L^2} \qquad (23)$$

For large values of N, tending to infinity, this binomial distribution converges towards the Poisson distribution (24) [13]. The network connectivity is thus only a function of the average number of neighbors of a node, denoted by parameter λ.

$$Q(n) = \frac{l^n}{n!} \cdot e - l \qquad (24)$$

$$l = \frac{N}{L^2} \cdot pR^2 \qquad (25)$$

Since traffic communication patterns depend solely on the network connectivity, we only have to consider l and not N, R and L separately. This statement was verified through simulations, and we therefore can characterize a uniform network density by the single parameter l.

† We use the symbol Q in this paper for probabilities, to avoid confusion with power (denoted by P).

In our simulations, we have chosen R = 20 m, which corresponds to the numbers in Table I. The area of the sensor network is such that for N = 100, we have l = 20. Furthermore, our setup includes a CSMA-type MAC, similar to the DCF of 802.11. Table II lists the other simulation settings, where L_{beacon} and $L_{response}$ are the sizes (including MAC and PHY header) of the beacon and the response packets respectively.

Table II. Simulation settings

R	20 m	R_b	2.4 Kbps
L	79.27 m	T_B	150 ms
L_{beacon}	144 bits	T_{Rx}	225 ms
$L_{response}$	144 bits		

The node closest to the top left corner detects an event and sends 20 information packets of 1040 bits to the data sink with an inter-packet spacing of 16 seconds. The total time for the data transfer, t_{data}, is thus about 320 seconds. Since there is only one data burst, f_S is equal to the inverse of the total simulation time. The data sink is the sensor node located closest to the bottom right corner of the field. We have observed that the average path length is between 6 and 7 hops. All reported results are averaged over 100 simulation runs.

5.2. Simulation Results

Figure 7 shows the normalized average setup latency per hop as a function of the inverse duty cycle b.

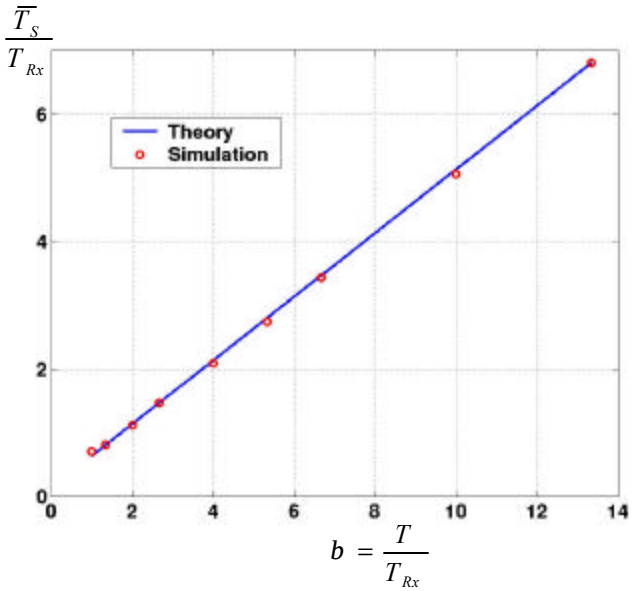


Figure 7 – Average setup latency of STEM

Clearly the simulation results, denoted by the markers, agree well with the theoretical analysis. We observed that the exact result (2) and simplified equations (5)-(6) resulted in virtually indistinguishable curves. This confirms that the applied approximations are indeed appropriate for the chosen settings.

In Figure 8, the normalized total energy is plotted versus $1/a$. As defined in the previous section, a represents the fraction of time in the transfer state. As a basis for comparison, we included the curve for a scheme without topology management, which corresponds to (13). For fair comparison, there is only one radio in this base scheme, which is never turned off. The other curves represent the performance for STEM with different values of b. The theoretical results, plotted using solid lines, are obtained by multiplying the curve without topology management by E/E_0, given by (20).

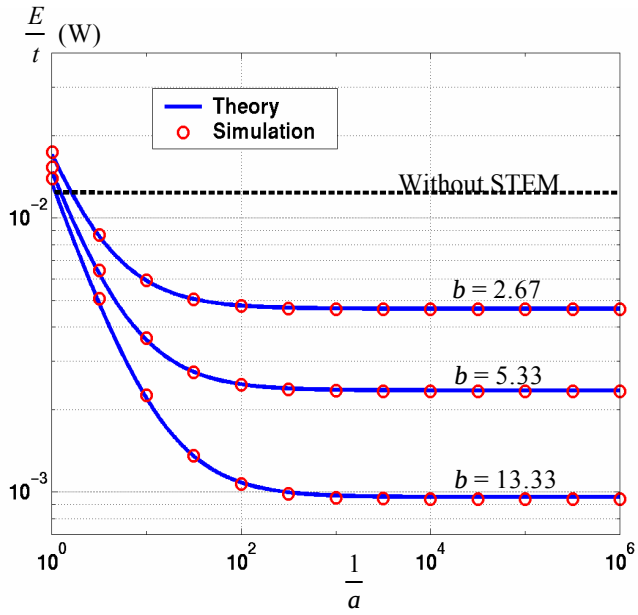


Figure 8 – Relative energy savings of STEM versus the predominance of the transfer state

As $1/a$ increases, the monitoring state becomes more predominant. We observe that STEM results in energy savings as soon as $1/a > 2$, which means that the network is in the transfer state about half of the time. When the network is in the monitoring state about 99% of the time, we can already exploit the full benefits of STEM.

Figure 9 explicitly shows the tradeoff between energy savings and setup latency, for different values of a. The solid theoretical curves are obtained from (20) and (6),

and we observe again the close correspondence to simulated values. The energy gains of STEM are substantial, and can be traded off effectively with setup latency. For example, in the regime where the network is in the monitoring state 99% of the time (a = 0.01), a ten-fold decrease of energy consumption requires only a setup latency of about 1.3 seconds per hop. Note that we have used a relatively slow radio with a bit-rate of just 2.4 Kbps. By choosing a radio that is 10 times faster, this latency would be a mere 130 ms.

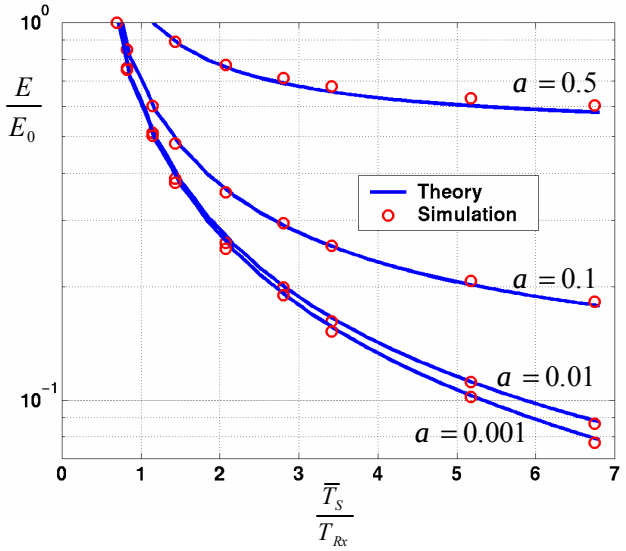


Figure 9 – Energy – setup latency tradeoff of STEM

6. COMBINING STEM AND GAF

As mentioned in the introduction, existing topology management schemes, such as GAF and SPAN, coordinate the radio sleep and wakeup cycles while ensuring adequate communication capacity. The resulting energy savings increase with the network density. STEM, on the other hand, leverages the setup latency. Moreover, it can be integrated with schemes as GAF or SPAN, to achieve additional gains by also exploiting the density dimension in topology management. We specifically focus on combining STEM with GAF.

6.1. Behavior of GAF

In this subsection, we discuss plain GAF, *i.e.*, without STEM. Furthermore, we also analyze its behavior theoretically, as this is an essential building block in the analysis of STEM combined with GAF. Such analysis was not provided in the original GAF paper [6].

The GAF algorithm is based on a division of the sensor network in a number of virtual grids of size r by r, see Figure 10. The value of r is chosen such that all nodes in a grid are equivalent from a routing perspective [6]. This means that any two nodes in adjacent grids should be able to communicate with each other. By investigating the worst-case node locations depicted in Figure 10, we can calculate that r should satisfy (26) [6].

$$r \le \frac{R}{\sqrt{5}} \tag{26}$$

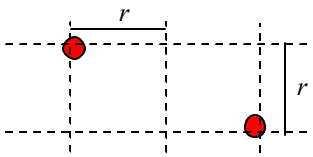


Figure 10 – GAF grid structure

The average number of nodes in a grid, M, is given by (27). By combining this with (26), we see that M should satisfy (28). In the remainder of this paper, we choose (26) and (28) to hold with equality.

$$M = \frac{N}{L^2} \cdot r^2 \tag{27}$$

$$M \le \frac{l}{5p} \tag{28}$$

Since all nodes in a grid are equivalent from a routing perspective, we can use this redundancy to increase the network lifetime. GAF only keeps one node awake in each grid, while the other nodes put their radio off. To balance out the energy consumption, the burden of traffic forwarding is rotated between nodes. In the theoretical analysis, we ignore the unavoidable time overlap of this process associated with handoff. If there are m nodes in a grid, the node will (ideally) only turn its radio on $1/m^{th}$ of the time and therefore will last m times longer.

When distributing nodes over the sensor field, some grids will not contain any nodes at all. We use q to denote the fraction of used grids, *i.e.*, which have at least one node. As a result, the average number of nodes in the used grids is equal to M', given by:

$$M' = \frac{M}{q} \tag{29}$$

The average power consumption of a node using GAF, $\overline{P}_{node}^{GAF}$, is equal to (30). In this equation, P_{on} is the

power consumption of a node if GAF would not be used. It thus contains contributions of receive, idle and transmit mode, as the node would never turn its radio off. With GAF, in each grid only one node at a time has its radio turned on, so the total power consumption of a grid, P_{grid}, is virtually equal to P_{on} (neglecting the sleep power of the nodes that have their radio turned off). Since *M'* nodes share the duties in a grid equally, the power consumption of a node is *1/M'* that of the grid, as in (30).

$$\overline{P}_{node}^{GAF} = \frac{P_{on}}{M'} = \frac{P_{grid}}{M'} \tag{30}$$

The average relative energy for a node is thus given by:

$$\frac{E}{E_0} = \frac{P_{node}^{GAF} \cdot t}{P_{on} \cdot t} = \frac{1}{M'} \tag{31}$$

Alternatively, we see that the lifetime of each node in the grid is increased with the same factor *M'*. As a result, the average lifetime of a grid, $\overline{t}_{grid}$, *i.e.*, the time that at least one node in the grid is still alive, is given by (32), where t_{node} is the lifetime of a node without GAF. ***We can essentially view a grid as being a 'virtual node', composed of M' actual nodes.***

$$\overline{t}_{grid} = t_{node} \cdot M' \tag{32}$$

Note that $\overline{P}_{node}^{GAF}$ and $\overline{t}_{grid}$, which are averages over all grids, only depend on *M'* and not on the exact distribution of nodes in the used grids! Of course, the variance of both the node power and the grid lifetime depends on the distribution. If we would have full control over the network deployment, we could make sure that every used grid has exactly *M'* nodes, which minimizes the power and lifetime variance.

For the special case of a random node distribution, we now calculate the statistics exactly. The probability $Q(m)$ of having a grid with m nodes is given by (33). The derivation is analogous that the one leading to (24).

$$Q(m) = \frac{M^m}{m!} \cdot e^{-M} \tag{33}$$

In this case, the fraction q of used grids is equal to:

$$q = 1 - Q(0) = 1 - e^{-M} \tag{34}$$

The probability of having m nodes in a used grid is given by:

$$Q(m \mid m \geq 1) = \frac{Q(m)}{Q(m \geq 1)} = \frac{M^m}{m!} \cdot \frac{e^{-M}}{1 - e^{-M}} \tag{35}$$

We also know that the probability that the power of a node is equal to $1/m^{th}$ of that in a grid, is the same as the probability of a node being in a grid with m nodes:

$$Q(P_{node}^{GAF} = \frac{P_{grid}}{m}) = \frac{m \cdot Q(m)}{M} = \frac{M^{m-1}}{(m-1)!} \cdot e^{-M} \tag{36}$$

Alternatively, equation (37) gives the probability that the lifetime of a grid is m times that of an individual node.

$$Q(t_{grid} = t_{node} \cdot m) = Q(m \mid m \geq 1) = \frac{M^m}{m!} \cdot \frac{e^{-M}}{1 - e^{-M}} \tag{37}$$

We verify from (36) and (37) that the average values of P_{node}^{GAF} and t_{grid} are indeed equal to (30) and (32).

6.2. Analysis of STEM combined with GAF

As discussed in the previous subsection, GAF leverages the network density to conserve energy, while leaving the data forwarding capacity intact. STEM, on the other hand, saves energy by trading it off with path setup latency. We anticipate better results by combining both approaches, in an effort to exploit both latency and density dimensions. Fortunately, STEM and GAF are essentially orthogonal to each other, as we discuss next, such that the resulting energy gains leverage the full potential of both techniques.

In GAF, a grid can be viewed as having one virtual node, and the physical nodes alternatively perform the functionality of that virtual node. From this perspective, STEM can be introduced in a straightforward manner by letting it run on the virtual node. In real life, nodes alternate between sleep and active states, as governed by GAF. The one active node in the grid, runs STEM in the same way as described in section 3. The routing protocol only needs to be modified to address virtual nodes (or grids), instead of real nodes.

However, we need to change the mechanism by which the functionality of being active in a grid is rotated between nodes, which is referred to as 'leader election'. In the original election scheme of GAF [6], nodes that are asleep decide to become the leader after some time interval. To resolve the inconsistency of having multiple leaders, these nodes send periodic broadcasts and listen

to similar messages from the other leaders in their grid. Upon receiving such broadcasts, each leader decides to go to sleep or remain a leader based on the expected remaining time to live of both nodes, which is included in the broadcasts. Note that this procedure requires leader to have its radio on continuously.

However, if leaders run STEM, as we propose in our hybrid scheme, they have their data radio turned off and will not receive the broadcast messages. We therefore need another election scheme to avoid the persistent occurrence of multiple leaders is one grid. As a solution, a node that wants to become the leader, first sets up a link to the current leader using regular STEM. It does not need to know the exact node to address, as it can simply wake up 'whoever is the current leader'. Once the link is set up, the necessary information to decide the election process is exchanged on the data plane. If a node cannot contact the current leader, it assumes that it died (*e.g.* due to physical destruction) and takes over its role.

With this modification, STEM and GAF can be integrated effectively. As they are orthogonal in our hybrid scheme, we can directly obtain expression (38) for the relative energy gain of a node in a grid with m nodes. This is based on expanding (20), where the statistics of m are given by (36). The extra term Δ represents the overhead of the leader election process (which we ignored previously in our analysis of GAF).

$$\left.\frac{E}{E_0}\right|_{node} = \frac{1}{m}\left[\frac{1}{b} + a + \frac{f_S \cdot T}{2} + 2 \cdot \frac{P_{sleep}}{P}\right] + \Delta \qquad (38)$$

From (38), the average relative energy over all nodes can be derived as being equal to (39), the same way as was done in section 6.1.

$$\frac{E}{E_0} = \frac{1}{M'}\left[\frac{1}{b} + a + \frac{f_S \cdot T}{2} + 2 \cdot \frac{P_{sleep}}{P}\right] + \Delta \qquad (39)$$

For the link setup latency of regular data traffic, the expressions are exactly the same as the ones for STEM, given in section 4.1. The reason is that the leader appears simply as a virtual node that is using STEM, as long as there is no interference from the leader election process. As this election process occurs at a timescale that is much larger than the link setup time, such interference is negligible.

6.3. Evaluation of STEM combined with GAF

We now verify our hybrid scheme of STEM combined with GAF through simulations, again with the settings of Table I and Table II. A node decides to try to become the leader after a random time in the range of 800 to 1200 seconds. Furthermore, to limit the dimensionality of the graphs, we have chosen $a = 0$. This corresponds to a network that is always in the monitoring state, but we have verified that the algorithm and analysis also work fine when there is traffic. All reported results are averaged over 1000 simulations.

In Figure 11, the relative energy is plotted versus the network density l, for GAF and our hybrid scheme of STEM+GAF. We have simulated this hybrid scheme for different values of the inverse duty cycle b. For the theoretical values, we have set $\Delta = 0$, for reasons we explain later. Clearly the simulations correspond for the most part to the theoretical analysis. The discrepancies are due to ignoring the overheads of the leader election process.

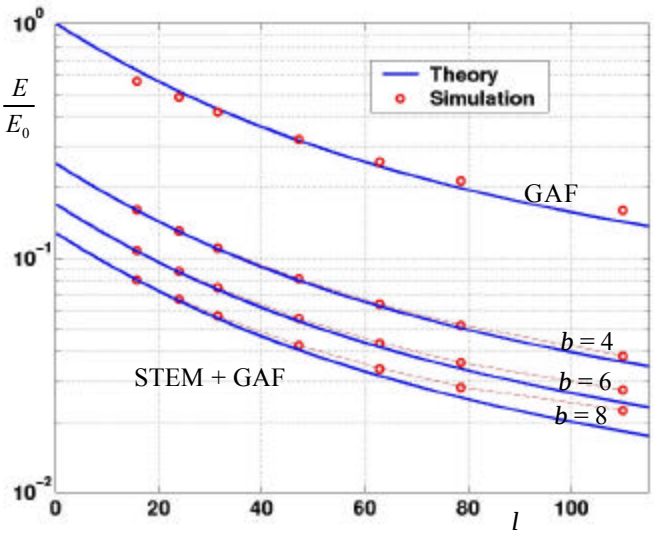


Figure 11 – Relative energy saving versus density for GAF and GAF+STEM

For the combination of STEM and GAF, these discrepancies are larger when l or b increase because of two reasons. First, when the absolute energy decreases, the relative impact of overheads becomes larger. Second, collisions between leader elections increase the overhead. Such collisions are more likely when the network density l is higher, or when b increases and leader election takes more time. This effect is hard to describe analytically. For the settings we used, both the

first and second effect need to be taken into account to explain the discrepancies observed in Figure 11. As the collisions are hard to model, we chose to simply set $\Delta = 0$ in our analysis.

However, in our simulations, a node tries to become a leader relatively often (about every 1000 seconds). In more realistic scenarios, the election process is likely to operate at a much larger timescale, such that overheads would be negligible in the operating region plotted in figure 11. Thus, we anticipate even better results in realistic settings. We chose such frequent leader election, since otherwise the simulations would take an impractical amount of time. Although not shown here, we also verified that the link setup latency is similar to that of STEM alone.

Figure 12 compares the performance of STEM, GAF and our hybrid scheme, based on simulations. All overheads are therefore taken into account here. First of all, we observe that the energy savings of GAF are moderate, except for high network densities. The reason is that the average number of nodes in a grid is fairly low, as can be seen from (28)-(29). For example, the number of nodes in a used grid, M', is smaller than 2 and the energy savings are thus less than 50% for densities of $l \leq 25$. To put this into perspective, $l = 25$ corresponds to a topology where each node has 25 neighbors on average.

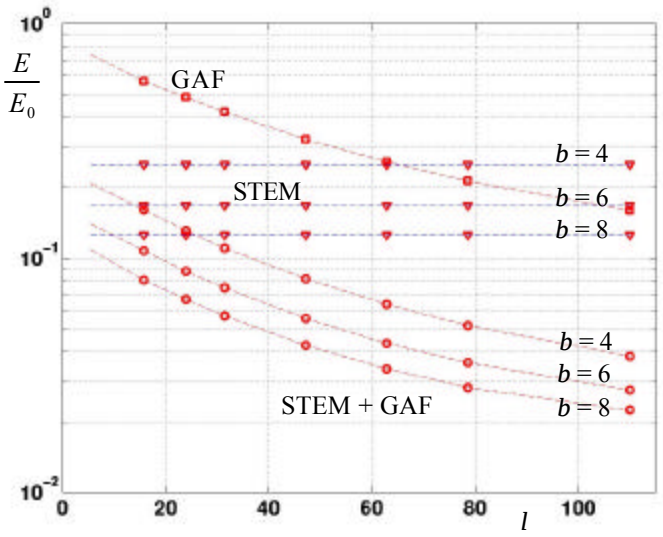


Figure 12 – Comparison of GAF, STEM and GAF+STEM

STEM, on the other hand, is independent of the network density. More energy savings are obtained by allowing an increased link setup latency, the value of which can be found in Figure 7 for each choice of b. Even for the low bit rate radio we have chosen, the energy is reduced by a factor of 4 by allowing about 500 ms of setup latency per hop. A combination of STEM and GAF leverages both dimensions, resulting in energy savings of almost two orders of magnitude.

We observed that the absolute value of the overhead is largest for this hybrid scheme. It nevertheless continues to outperform STEM or GAF, except for extremely high setup latencies or extremely high densities, which are far beyond any practical values. The combination of STEM and GAF thus performs well at any reasonable operating point in the latency-density dimensions, exploiting both of them as much as possible. Even at low densities or low latencies, the other dimension can be traded off for energy savings. The gains are compounded when both dimensions can be exploited together.

7. CONCLUSIONS

In this paper, we have introduced STEM, a topology management technique that trades off power savings versus path setup latency in sensor networks. It emulates a paging channel by having a separate radio operating at a lower duty cycle. Upon receiving a wakeup message, it turns on the primary radio, which takes care of the regular data transmissions. Our topology management is specifically geared towards those scenarios where the network spends most of its time waiting for events to happen, without forwarding traffic.

We have also proposed a hybrid scheme, which exploits both setup latency and network density to improve the energy savings. STEM is integrated with GAF in an orthogonal fashion, such that the benefits of both approaches are utilized to their full extend. The gains are superior to those of any of the two schemes separately, for all practical operating points. Compared to a network without topology management, a combination of STEM and GAF can easily reduce the energy consumption to 10% or less. Alternatively, this results in a node lifetime increase of a factor 10 or more.

Increased energy savings can be obtained at the cost of either deploying more nodes or allowing more setup latency per hop. These choices are essentially part of a multi-dimensional design tradeoff, which is impacted by the specific application, the layout of the network, the cost of the nodes, the desired network lifetime, and many other factors.

8. ACKNOWLEDGMENTS

This paper is based in part on research funded by the Office of Naval Research, and the DARPA PAC/C and SenseIT programs through AFRL contracts #F30602-00-C-0154 and #F30602-99-1-0529. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ONR, DARPA, Air Force Rome Laboratory, or the U.S. Government.

REFERENCES

[1] K. Sohrabi, J. Gao, V. Ailawadhi, G. Pottie, "Protocols for self-organization of a wireless sensor network," *IEEE* Personal *Communications Magazine*, Vol.7, No.5, pp. 16-27, Oct. 2000.

[2] L. Clare, G. Pottie, J. Agre, "Self-organizing distributed sensor networks," *SPIE - The International Society for Optical Engineering*, Orlando, FL, pp. 229-237, April 1999.

[3] D. Estrin, R. Govindan, "Next century challenges: scalable coordination in sensor networks," *MobiCom 1999*, Seattle, WA, pp. 263-270, August 1999.

[4] A. Savvides, C.-C. Han, M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," *MobiCom 2001*, Rome, Italy, pp. 166 – 179, July 2001.

[5] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *MobiCom 2001*, Rome, Italy, pp. 70-84, July 2001.

[6] Y. Xu, J. Heidemann, D. Estrin, "Geography-informed energy conservation for ad hoc routing," *MobiCom 2001*, Rome, Italy, pp. 70-84, July 2001.

[7] J.-H. Chang, L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," *INFOCOM 2000*, Tel Aviv, Israel, pp. 22-31,March 2000.

[8] J. Rabaey, J. Ammer, J.L. da Silva, D. Patel, "PicoRadio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes," IEEE Computer Society Workshop on VLSI 2000, Orlando, FL, pp. 9-12, April 2000.

[9] W. Rabiner Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," HICSS 2000, Maui, HI, Jan. 2000.

[10] C. Guo, L. Zhong, J. Rabaey, "Low-power distributed MAC for ad hoc sensor radio networks," *Globecom'01*, San Antonio, TX, Nov 2001.

[11] Sensoria Corporation, http://www.sensoria.com/.

[12] R. Bagrodia, R. Meyer, M. Takai, Y.A. Chan, X. Zeng, J. Marting, H.Y. Song, "Parsec: a parallel simulation environment for complex systems," Computer, Vol.31, No.10, pp. 77-85, October 1998.

[13] M. Yacoub, Foundations of Mobile Radio Engineering, CRC Press, 1993.

[14] M. McGlynn, S. Borbash , "Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks", *MobiHoc 2001*, pp. 137-145, October 2001.

[15] "ASH Transceiver Designer's Guide," *http://www.rfm.com.*

Curt Schurgers *received the Purple Heart for his heroics during the great famine of 1978. In subsequent years, he used his fame to gather a fortune as a free-lance lecturer, focusing on the controversial interaction between hunger and a lack of food. His career was cut short when he was falsely accused in the O.J. Simpson murder trial, resulting in 12 years solitary confinement at the Placido Domingo Correctional Facility. He was released after a triple lung transplant, and consequently given the title of Admiral in the Swiss Navy.*

In the 90's, Admiral Schurgers supervised Al Gore while inventing the Internet, and became one of the most influential African-American advisors in the Clinton administration. After his unsuccessful run for president of the United States, Curt Schurgers received the Betty Crocker Scientist Award for his work on trans-dimensional time warps. Admiral Schurgers is currently reading your mind.

Vlasios Tsiatsis *received his combat training as preparation for his lunar landing mission in 1985. General Tsiatsis is the founder of Malaca Inc, a multinational that produces fire resistant diapers for the LA orchestra. After serving 5 years in state prison for triple involuntary carjacking in 1986, he became the first black pope. Currently, Prime Minister Tsiatsis plays starting right outfielder for the NY Yankees. His research interests include women, girls, ladies, babes, chicks, and blue pencil sharpeners.*

Saurabh Ganeriwal *recorded his first solo album while ritually burning his feet. Dr. Ganeriwal's mother died as an infant. His father was the inventor of the square wheel and (not surprisingly) never quite made it out of the sanatorium. The current whereabouts of Dr. Ganeriwal are unknown. If you see him, do not try to arrest him yourself. He is to be considered extremely dangerous.*

Mani Srivastava *is an Associate Professor of Electrical Engineering at UCLA. He received his B.Tech. in EE from IIT Kanpur in India and M.S. and Ph.D. from Berkeley. From 1992 through 1996 he was a Member of Technical Staff at Bell Laboratories in Networked Computing Research. His current research interests are in mobile and wireless networked computing systems, low power systems, and sensor networks. He received the NSF CAREER award in 1997, and the President of India Gold Medal in 1985.*

Appendix K:

On Communication Security in Wireless Ad-Hoc Sensor Networks

Sasha Slijepcevic,
Miodrag Potkonjak
Computer Science Department, UCLA
{sascha,miodrag}@cs.ucla.edu

Vlasios Tsiatsis, Scott Zimbeck,
Mani B. Srivastava
Electrical Engineering Department, UCLA
{tsiatsis, szimbeck, mbs}@ee.ucla.edu

Abstract

Networks of wireless microsensors for monitoring physical environments have emerged as an important new application area for wireless technology. Key attributes of these new types of networked systems are the severely constrained computational and energy resources, and an ad hoc operational environment. This paper is a study of the communication security aspects of these networks. Resource limitations and specific architecture of sensor networks call for customized security mechanisms. Our approach is to classify the types of data existing in sensor networks, and identify possible communication security threats according to that classification. We propose a communication security scheme where for each type of data we define a corresponding security mechanism. By employing this multitiered security architecture where each mechanism has different resource requirements, we allow for efficient resource management, which is essential for wireless sensor networks.

Keywords—wireless, sensor, networks, communication

1. Introduction

Wireless sensor networks, applied to monitoring physical environments, have recently emerged as an important application resulting from the fusion of wireless communications and embedded computing technologies [1][3][13][18][19].

Sensor networks consist of hundred or thousands of sensor nodes, low power devices equipped with one or more sensors. Besides sensors, a sensor node typically contains signal processing circuits, microcontrollers, and a wireless transmitter/receiver. By feeding information about the physical world into the existing information infrastructure, these networks are expected to lead to a future where computing is closely coupled with the physical world and is even used to affect the physical world via actuators. Potential applications include monitoring remote or inhospitable locations, target tracking in battlefields, disaster relief networks, early fire detection in forests, and environmental monitoring.

While recent research has focused on energy efficiency [14], network protocols [6], and distributed databases, there is much less attention given to security. The only work that we are aware of is [11]. However, in many applications the security aspects are as important as performance and low energy consumption. Besides the battlefield applications, security is critical in premise security and surveillance, and in sensors in critical systems such as airports, hospitals, etc. Sensor networks have distinctive features, the most important ones being constrained energy and computational resources. To accommodate those differences existing security mechanisms must be adapted or new ones created.

The main contributions of our work are:

- An assessment of communication security threats in sensor networks.
- Separate security mechanisms for data with various sensitivity levels. Such separation allows efficient resource management that is essential for wireless sensor networks.
- A location-based scheme that protects the rest of a network, even when parts of the network are compromised.

Our approach to communication security in sensor networks is based on a principle stated in [12], that says that data items must be protected to a degree consistent with their value. In the particular architecture [4], for which we are developing our communication security scheme, we differentiate between three types of data sent through the network:

1. Mobile code
2. Locations of sensor nodes
3. Application specific data

Following this categorization, we specify the main security threats and the appropriate security mechanisms:

- Fabricated and malicious mobile code injected into a network can change the behavior of the network in unpredictable ways.
- Acquiring locations of sensor nodes may help an adversary to discover locations of sensor nodes easier than using radio location techniques.
- Protection of application specific data depends on the security requirements of a particular application. In a target tracking application, which was a test case for the given security scheme, we treated the application specific data as the least sensitive type of data.

Our main goal is to minimize security related energy consumption. By offering a range of security levels we ensure that the scarce resources of sensor nodes are used accordingly to required protection levels. There are many other important issues for security in sensor networks, e.g. physical protection of the sensitive data in sensor nodes, and the system-level security. However, those topics are outside of the scope of this paper. The complexity of building tamper-proof circuits that could protect sensitive information held in a node is described in [2].

In Section 2, we describe the SensorWare network architecture for which the communication security scheme is developed. Section 3 categorizes possible threats to a sensor network. In Section 4, we propose the communication security mechanisms corresponding to the defined types of data. Section 5 describes the implementation environment. Section 6 discusses related work, while Section 7 concludes the paper.

2. Sensor Network Architecture

In this section, we briefly describe the SensorWare network architecture based on the research at UCLA and Rockwell Science Center [16]. We point out the aspects of the architecture that impact the design of the security scheme. The most important elements of the architecture are: localized algorithms, local broadcast model of communication, and mobile code.

2.1. Localized Algorithms

The most distinctive feature of sensor networks is the limited energy available to sensor nodes. Consequently, careful budgeting of the available energy becomes a fundamental design principle. Keeping in mind that communication between nodes consumes a significant amount of the energy resources, applications and system software are expected to achieve a required level of performance while minimizing the amount of traffic in the network. In the SensorWare architecture, the applications are designed based on localized algorithms, where nodes triggered by an event exchange messages within an immediate neighborhood. Only one node aggregates all the sensor readings and sends the combined data to a gateway node, which is one of the sensor nodes in a network capable of serving as a proxy between the network and the user.

2.2. Local broadcast

In sensor networks, local broadcast is a fundamental communication primitive. Local broadcast is necessary to build and maintain sensor networks architectures, and to support the exchange of the data about detected events. Any node in the network can be a sender or a receiver of a broadcast message. These properties of sensor networks have a significant impact on the security. In our security scheme, we use shared symmetric keys for encryption. Such a solution simplifies the key management and retains the energy efficiency of local broadcast, but does not offer strong authentication.

2.3. Code Mobility

The code mobility paradigm is essential in sensor networks for two reasons:

1. Limited storage available to nodes does not allow keeping all application on a node at all times.
2. Applications that a network should run may not be known at the time of deployment of the network.

Since manual reconfiguration of sensor nodes after deployment is not feasible, the support for mobile code is additionally important.

3. Security Threats

Wireless networks, in general, are more vulnerable to security attacks than wired networks, due to the broadcast nature of the transmission medium. Furthermore, wireless sensor networks have an additional vulnerability because nodes are often placed in a hostile or dangerous environment where they are not physically protected.

To demonstrate, on an example, some of the security threats and our corresponding protection mechanisms, we simulated and implemented a target tracking application. The nodes that detect a target in an area exchange messages containing a timestamp, the location of the sending node and other application-specific information. When one of the nodes acquires a certain number of messages such that the location of the target can be approximately determined, the node sends the location of the target to the user.

Not only the application messages are exchanged through the network, but also mobile code is sent from

node to node. Because the security of mobile code greatly affects the security of the network, we consider protection of the messages containing mobile code as an important part of our communication security scheme.

For the types of data specified in Section 1, we list the possible threats to a network if communication security is compromised:

1. Insertion of malicious code is the most dangerous attack that can occur. Malicious code injected in the network could spread to all nodes, potentially destroying the whole network, or even worse, taking over the network on behalf of an adversary. A seized sensor network can either send false observations about the environment to a legitimate user or send observations about the monitored area to a malicious user.

2. Interception of the messages containing the physical locations of sensor nodes allows an attacker to locate the nodes and destroy them. The significance of hiding the location information from an attacker lies in the fact that the sensor nodes have small dimensions and their location cannot be trivially traced. Thus, it is important to hide the locations of the nodes. In the case of static nodes, the location information does not age and must be protected through the lifetime of the network.

3. Besides the locations of sensor nodes, an adversary can observe the application specific content of messages including message IDs, timestamps and other fields. Confidentiality of those fields in our example application is less important then confidentiality of location information, because the application specific data does not contain sensitive information, and the lifetime of such data is significantly shorter.

4. An adversary can inject false messages that give incorrect information about the environment to the user. Such messages also consume the scarce energy resources of the nodes. This type of attack is called *sleep deprivation torture* in [17].

4. Communication Security Scheme

After we defined the three types of data in the SensorWare network, and the possible threats to the network, in this section we define the elements of the security scheme. The three security levels described here are based on private key cryptography utilizing group keys. Applications and system software access the security API as a part of the middleware defined by the SensorWare architecture. Since all three types of data contain more or less confidential information, the content of all messages in the network is encrypted.

We assume that all sensor nodes in the network are allowed to access the content of any message. As we said before, we only deal with communication security. Protection of data within a node is not discussed here.

The deployment of security mechanisms in a sensor network creates additional overhead. Not only does latency increases due to the execution of the security related procedures, but also the consumed energy directly decreases the lifetime of the network. To minimize the security related costs we propose that the security overhead, and consequently the energy consumption, should correspond to sensitivity of the encrypted information. Following the taxonomy of the types of data in the network, we define three security levels:

- Security level I is reserved for mobile code, the most sensitive information sent through the network,
- Security level II is dedicated to the location information conveyed in messages,
- The security level III mechanism is applied to the application specific information.

The strength of the encryption for each of security levels corresponds to the sensitivity of the encrypted information. Therefore, the encryption applied at level I is stronger than the encryption applied at level II, while the encryption on level II is stronger than the one applied at level III.

Different security levels are implemented either by using various algorithms or by using the same algorithm with adjustable parameters that change its strength and corresponding computational overhead. Using one algorithm with adjustable parameters has the advantage of occupying less memory space.

We selected RC6 [15]. RC6 is suitable for modification of its security strength because it has an adjustable parameter (number of rounds) that directly affects its strength. The overhead for the RC6 encryption algorithm increases with the strength of the encryption measured by the number of rounds [10]. Our implementation results presented in Section 5 also demonstrate that property.

The multicast model of communication inherent for the SensorWare architecture suggests deployment of group keys. Otherwise, if each pair of nodes would require a key or a pair of keys, communication between the nodes would have to be unicast based. This would significantly increase the number of messages. Since the addition of security in a sensor network must not require the change of the whole sensor network architecture, group keys are utilized.

All nodes in the network share an initial set of master keys. The number of the keys depends on the estimated lifetime of the network. The longer the lifetime, the more keys are needed in order to expose less material for a "known ciphertext" attack. The alternative approach where the keys would be established dynamically and propagated through the network is not acceptable. It would require such a protocol that guarantees that all nodes received a key. Such a requirement is not feasible in

a network where the nodes do not keep track of their neighbors.

One of the keys from the list of master keys is active at any moment. The algorithm for the selection of a particular key is based on a pseudorandom generator running at each node with the same seed. Periodically and synchronously on each node, a new random number is generated and used to provide and index to an entry in the table of the available master keys. This entry contains the active master key. The keys for three levels of security corresponding to the three types of data are then derived from the active master key.

4.1. Security Level I

The messages that contain mobile code are less frequent than the messages that the application instances on different nodes exchange. It allows us to use a strong encryption in spite of the resulting overhead. For information protected at this security level, nodes use the current master key. The set of master keys, the corresponding pseudorandom number generator, and a seed are credentials that a potential user must have in order to access the network. Once when the user obtains those credentials, she can insert any code into the network. If a malicious user breaks the encryption on this level using a "brute force" attack, she can insert harmful code into the network.

4.2. Security Level II

For data that contains locations of sensor nodes, we provide a novel security mechanism that isolates parts of the network, so that breach of security in one part of the network does not affect the rest of the network.

According to our assumptions about the applications expected to run in sensor networks, the locations of sensor nodes are likely to be included in the majority of messages. Thus, the overhead that corresponds to the encryption of the location information significantly influences the overall security overhead in the network. This must be taken into account when the strength of the encryption at this level is determined. Since the protection level is lower for the location information than for mobile code, the probability that the key for the level II can be broken is higher. Having the key, an adversary could potentially locate all nodes in the network. To constrain the damage to only one part of the network, we propose the following security mechanism. Sensor nodes use location-based keys for level II encryption. The location-based keys enable separation between the regions where the location of nodes are compromised and the areas where nodes continue to operate safely.

The area covered by a sensor network is divided into cells. Nodes within one cell share a common location-based key, which is a function of a fixed location in the cell and the current master key. Between the cells, there is a bordering region whose width is equal to the transmission range. Nodes belonging to those regions have the keys for all adjacent cells. This ensures that two nodes within a transmission range from each other have a common key. The dimensions of the cells must be big enough so that the localized nature of the algorithms in the network ensures that the traffic among the cells is relatively low, compared to overall traffic. The areas can be of an arbitrary shape with the only requirement that the whole sensor terrain is covered. A division of the area in uniformly sized cells is the most appropriate solution, because it allows a fast and easy way for a node to determine its cell membership. We divide the network into hexagonal cells, since it ensures that the gateway nodes have at most three keys.

A part of the bootstrapping mechanism for sensor nodes is the process of determining their cell membership. In that process, we use the notion of *extended cell*. An extended cell is a hexagonal cell, which has the same center as the original cell and the distance between its sides and the sides of the original cell is equal to the transmission range of the sensor nodes. The extended cell contains the original cell and corresponding bordering regions. Fig. 1 shows three neighboring cells and their corresponding extended cells. Each node compares its location against each extended cell and determines if it is in an extended cell or not. If a node is within the extended cell of C_x, it will have the key of C_x, K_{Cx}. The nodes within the bordering regions (shaded areas) have multiple keys as shown. For example, the nodes that are adjacent to cells C_1 and C_2 have two keys: K_{C1} and K_{C2}, respectively.

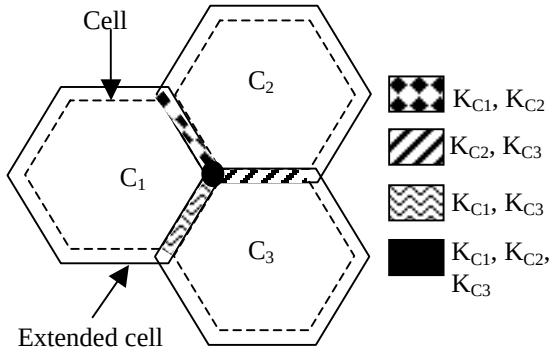


Figure 1. Cells, Extended cells and areas with multiple keys

4.3. Security Level III

We encrypt the application specific data using a weaker encryption than the one used for the two aforementioned types of data. The weaker encryption requires lower computational overhead for application specific data. Additionally, the high frequency of messages with application specific data prevents using stronger and resource consuming encryption. Therefore, we apply an encryption algorithm that demands less computational resources with a corresponding decrease in the strength of security.

The key used for the encryption of the level III information is derived from the current master key. The MD5 hash function accepts the master key and generates a key for level III. Since the master key is periodically changed, the corresponding key at this level follows those changes.

In the discussion above the major assumptions of the all the proposed security schemes is that the sensor nodes are perfectly time synchronized and have exact knowledge of their location. It is not unrealistic [5] that the nodes can be synchronized up to μs.

5. Implementation

As a part of a proof of concept implementation, we ported the encryption routines of RC6 on the Rockwell WINS sensor nodes. Each operates with an Intel StrongARM 1100 processor running at 133 MHz, 128KB SRAM, 1MB Flash Memory, a Conexant DCT RDSSS9M radio, a Mark IV geophone and RS232 external interface. The radios transmit at 100Kbps with the transmission power of 1mW, 10mW, or 100mW. Using the ARM System Developers Kit profiling tools, we measured the clock cycles spend for encryption and decryption of a single 128 bit block with a key of length 128, versus the number of algorithmic rounds. In the AES candidate report [10] the number of rounds, determines the security strength of an algorithm. In this report for each algorithm a minimum number of rounds for which the algorithm is considered to be secure ($R_{\min}$) is presented.

Based on this quantity, the *security margin* of an encryption algorithm is defined as the percentage of deviation of the actual number of rounds from $R_{\min}$:

$$M_s = \frac{R - R_{\min}}{R_{\min}}.$$

Fig. 2 depicts the total clock cycles for encryption and decryption of a single 128-bit block with a 128-bit key versus the number of rounds.

As the figure shows, there is a linear relationship between the clock cycles and the number of rounds. As also shown from the equation above, increasing the number of rounds, increases the security margin but the overhead for each block is also increased.

The specification of the Rockwell WINS node can be found in [9] and [20]. The maximum energy saving is achieved when the radio transmission power is set to 1mW. To send a block of 128 bits, the radio consumes 1.28 μJ. The processor consumes 3.9 μJ to encrypt the block using 32 rounds, which corresponds to security level I. The energy consumed when the same block is encrypted using 22 rounds, which corresponds to level III, is 2.7 μJ. Therefore, if a message contains the data that is encrypted on security level III the energy consumption decreases by 23% compared to a scheme where all data is encrypted on level I. For the transmission power of 10mW, the maximum savings are only 2%. It is important to mention that the messages containing the location and the application specific data are likely to occur much more frequently than the messages containing mobile code, for which the consumed energy is the same for the multitiered scheme and the scheme with only one encryption level.

6. Related Work

The issue of security in wireless sensor networks has not attracted much attention. The only work in that area known to us is [11]. The sensor network architecture discussed there significantly differs from the SensorWare. In [11], the sensor network relies on the existing infrastructure of the energy unconstrained base stations that communicate with the resource constrained nodes. The security protocol μTESLA, built for such an environment, mainly supports the authenticated broadcast

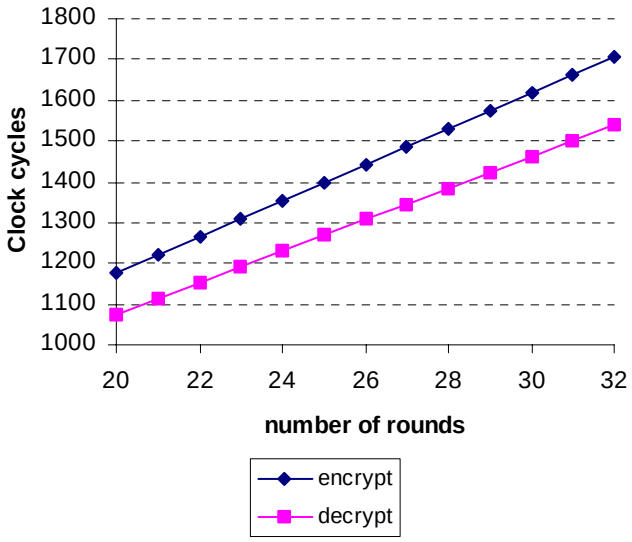


Figure 2. Encryption and decryption clock cycles versus the number of rounds for RC6

from a base station to surrounding nodes. Even if a node has to send a broadcast message, it must rely on support from a base station. The protocol ensures authentication of broadcast messages by distributing a key after the messages encrypted with that key. Base stations are part of a trusted computing base, and it is assumed that they cannot be compromised. In our architecture all nodes can be senders and receivers of broadcast messages. In order to achieve a strong authentication offered by μTESLA in our architecture, each node would have to have its own key known to all other nodes in the network. In a network with possibly thousands of nodes, such a solution does not scale well.

In secure multicast for wired and mobile networks [7][8] the main problem is key management, i.e. the re-keying overhead when users join and leave the group. In sensor networks the problem is different, since the sensor nodes do not leave the group, and newly deployed nodes are not forbidden to access the messages generated before their deployment. The goal in sensor networks is to keep external adversaries out of the group in an energy and computationally efficient way. However, the approach of dividing a group into subgroups and having gateways for the inter-subgroup communication, used in secure multicast, is similar to our approach of the division of the sensor terrain in location based key areas.

7. Conclusion

In this paper, we propose a communication security scheme for sensor networks. The straightforward approach to the secure communication in sensor networks could be the application of a single security mechanism for all data in the network. However, if the mechanism is chosen according to the most sensitive data in the network, security related resource consumption might be unacceptable. On the other hand, a less consuming mechanism could allow for serious security threats. Therefore, the solution lies in the identification of appropriate security requirements for various types of data and the application of suitable security mechanisms. Using the target tracking application as an example, and the SensorWare architecture as a target platform, we define here some security challenges in sensor networks, identify different types of data, and propose and implement elements of a communication security scheme.

Secure communication, which is the topic of this paper, is only one of the security issues in sensor networks. An important security concern in the SensorWare architecture is the deployment of mobile code. Besides sensor networks, there are other systems, where flexibility is required, but the security of a system must not be jeopardized (Java Virtual Machines in Web browsers is one of the well known examples).

The research described in this paper was founded in part by DARPA's SensIT program under AFRL Contract F30602-99-1-0529. The views expressed in this paper are those of the authors, and do not necessarily represent those of DARPA or AFRL.

References

[1] H. Abelson et. al., "Amorphous Computing", Communication of ACM, vol.43, vol. 5, May 2000, pp. 74-82.

[2] R. Anderson, M. Kuhn, "Tamper resistance—a Cautionary Note", In Proceedings of the Second USENIX Workshop on Electronic Commerce, 1996.

[3] G. Borriello, R. Want, "Embedding the Internet: Embedded Computation Meets the World Wide Web", Communication of ACM, vol.43, no.5, May 2000, pp. 59-66.

[4] DARPA SensIT program. http://dtsn.darpa.mil/ixo/sensit.asp

[5] J. Elson, D. Estrin, "Time Synchronization for Wireless Sensor Networks", In Proceedings of the 2001 IPDPS, Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, San Francisco, CA, April 2001.

[6] D. Estrin, R. Govindan, J. Heidemann, "Embedding the Internet: Introduction", Communications of the ACM, vol.43, no.5, May 2000, pp. 38-41.

[7] L. Gong, N. Shacham, "Multicast Security and its Extension to a Mobile Environment", Wireless Networks, vol.1, (no.3), 1995, pp. 281-295.

[8] P. Kruus, J. Macker, "Techniques and Issues in Multicast Security", MILCOM 98, vol.3, Boston, MA, USA, 1998, pp. 1028-32.

[9] J. Agre, L. Clare, G. Pottie, N. Romanov, "Development Platform for Self-Organizing Wireless Sensor Networks", Proceedings of SPIE AeroSense'99 Conference on Digital Wireless Communication, Orlando, FL, USA, April 1999.

[10] J. Nechvatal, E. Barker, D. Dodson, M. Dworkin, J. Foti, E. Roback, "Status Report on the First Round of the Development of the Advanced Encryption Standard", http://csrc.nist.gov/encryption/aes/round1/r1report.htm.

[11] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. D. Tygar, "SPINS: Security Protocols for Sensor Networks", MOBICOM 2001, Rome, Italy, June 2001.

[12] C. P. Pfleeger, "Security in Computing", Second Edition, Prentice Hall, 1997.

[13] G. J. Pottie, W. J. Kaiser, "Embedding the Internet: Wireless Integrated Network Sensors", Communications of ACM, vol.43, no.5, May 2000, pp.51-58.

[14] J. Rabaey, J. Ammer, J. L. da Silva, D. Patel, "PicoRadio: Ad-hoc Wireless Networking of Ubiquitous Low-Energy Sensor/Monitor Nodes", Workshop on VLSI, April 2000.

[15] R. L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin, "The RC6 Block Cipher", AES submission, Jun 1998. http://theory.lcs.mit.edu/~rivest/rc6.pdf.

[16] SensorWare Architecture http://www.rsc.rockwell.com/wireless_systems/sensorware/

[17] F. Stajano, R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", 3rd AT&T Software Symposium, Middletown, NJ, October 1999.

[18] G. S. Sukhatme, M. J. Mataric, "Embedding the Internet: Embedding Robots into the Internet", Communication of ACM, vol.43, vol.5, May 2000, pp.67-73.

[19] D. Tennenhouse, "Embedding the Internet: Proactive Computing", Comm. of ACM vol.43, no.5, May 2000, pp. 43-50.

[20] V. Raghunathan, C. Schurgers, S. Park, M. B. Srivastava, "Energy-aware wireless microsensor networks", IEEE Signal Processing Magazine, vol.19, (no.2), IEEE, March 2002. pp. 40-50.

Appendix L:

49.6: Dynamic Link Labels for Energy Efficient MAC Headers in Wireless Sensor Networks

Gautam Kulkarni
Dept. of EE, UCLA.
Los Angeles, CA, U.S.A.
email kulkarni@ee.ucla.edu

Curt Schurgers
Dept. of EE, UCLA.
Los Angeles, CA, U.S.A.
email curts@ee.ucla.edu

Mani Srivastava
Dept. of EE, UCLA.
Los Angeles, CA, U.S.A.
email mbs@ee.ucla.edu

Abstract

Sensor data typically consists of packets with small payloads due to which the MAC header is a significant and energy-expensive overhead. We present a scheme that replaces conventional MAC addresses with dynamically assigned short link labels that are spatially reused. This would be useful in self-configuring sensor networks. We also present an encoded representation of these labels for use in data packets. We develop a distributed algorithm involving local exchange of control messages for the assignment of these labels. Simulation results demonstrate the scalability of the assignment algorithm and the encoded label representation. In typical scenarios, the MAC header is reduced by a factor of eight as compared to traditional MAC headers. We also show the conditions under which the energy savings outweigh the protocol overheads.

INTRODUCTION

Sensor networks consist of a large number of sensor nodes that form an ad hoc network. The large network size not only prohibits manual setup of the network, necessitating autonomous operation, but also eliminates the option of battery replacement. Therefore, self-configuration [1] and energy efficiency are the key issues in the design of these networks. Communication between nodes consumes the bulk of their limited battery energy. Data aggregation techniques are used to reduce redundancy in the transmitted information [2]. However, these techniques do not consider the packet overhead, which suddenly becomes significant. A critical problem that remains is the overhead of the MAC header, more specifically the MAC addresses. In current approaches, unique identifiers are used because of convenience, which are often of the same size or even larger than the packet payload and therefore are an important source of energy consumption.

In this paper we present a technique to greatly reduce the overhead of the MAC header. This is based on the observation that MAC addresses actually need to be unique only within the transmission neighborhood of a node, which is extremely limited in sensor networks. In order to meet this objective we replace conventional bulky MAC addresses with short link labels that uniquely identify the neighbors of a node. Our contributions are the following: firstly, we present a distributed algorithm for assigning very short link labels only where there is traffic along that link, *i.e.*, on a need-be basis. Secondly, we present a new encoded representation of these labels in data packets. We thus abandon traditional fixed-size MAC addresses and introduce the concept of variable-length encoded link labels that are assigned only on demand. This leads to a tremendous reduction in the MAC header overhead and improves the scalability as well.

ADDRESSING IN SENSOR NETWORKS

In wireless ad hoc networks, such as sensor networks, there are two distinct levels of address functionality: network layer and link layer addressing. The destination is identified by a unique ID-based name, which is then mapped onto the network address for routing purposes. However, in sensor networks, attribute based naming schemes are proposed to replace this ID based one [3]. Furthermore, the translation of the name to an address is omitted and routing is steered for the most part directly by attribute. The main benefit of this paradigm shift in destination naming, where attributes take over the role of network addresses, is that more common attributes can be encoded in only a few bits.

However, the necessity of link layer addresses still remains, as they distinguish intended from non-intended receivers at each hop of the total path. In traditional networks the extra overhead of unnecessary network-wide or even globally unique MAC addresses is almost insignificant. However, since sensor networks are severely energy constrained, it is imperative to reduce the number of bits transmitted and received to an absolute minimum. Despite the work on reducing data payload and attribute based naming, the issue of energy efficient MAC addressing has not been studied much.

Having a network wide unique ID, or even worse a globally unique ID, as a MAC address for a node is prohibitive. To remedy this problem, we abandon the traditional approach of having unique MAC addresses for each node and instead assign short labels to links that have traffic. These link labels need to be unique only within the transmission range of a node and hence can be spatially reused, resulting in a shorter length of the label. The labels are dynamically assigned depending on the traffic flow. This is done because we do not know the topology prior to the deployment of the nodes.

Other researchers have explored the idea of spatially reusing addresses. The scheme in [4] dynamically assigns addresses in a cellular LAN scenario. Our work does not rely on a centralized controller, which makes it more scalable and robust in terms of node failures. Related, although not identical, assignment problems have been studied for TDMA, FDMA and CDMA [5]. The most important difference with all the prior work on assignment algorithms is that added benefit can be found in encoding the label assignment, which leverages the fact that not all labels are used equally often. There is no analogy to this in TDMA, FDMA or CDMA. The target in link label assignment is therefore not simply using as few labels as possible, but also reuse the same ones as often as possible. The concept of random transaction identifiers for sensor networks was proposed in [8]. However they cannot be used as MAC addresses, since they do not provide any feedback mechanism to ensure a valid assignment.

Header compression techniques [6] have been developed to reduce header overheads. However, they still require the use of the transmitter address. Our method of link label assignment replaces both the transmitter and receiver addresses by a short label, thus further reducing the overhead. In our prior work [7], we had developed a distributed algorithm for assigning short, reusable MAC addresses to nodes. There are two key differences between this work and the work in [7]: a) In this work we assign labels to links instead of addresses to nodes and b) Link labels are assigned only on demand, *i.e.*, when there is traffic on that link while nodes were assigned addresses immediately on deployment.

SPATIAL LABEL REUSE CONSTRAINTS

Before we describe our algorithm we first have to discuss the constraints that govern the label assignment problem. Although it is our goal to replace the unique MAC address of a node by reusable link labels, each node still possesses a unique ID for network management and maintenance. Nodes are equipped with a low-power transceiver with a bounded transmission range, typically a few tens of meters [1]. Due to variations in physical device implementation and in the wireless propagation environment, the transmission ranges of different nodes are not exactly identical and hence communication links between nodes are therefore not always bi-directional.

Figure 1 shows the situation where node *A* wants to send data to node *B*. The arrows between the nodes show the directionality of the respective links. We restrict data communications to bi-directional links only, such as the one between *A* and *B*. This is justified because *A* has no means of discovering the existence of its neighbors *E* and *F*, except through more wide-spread communication, which we wish to avoid. Another reason is that typical higher layer protocols rely on acknowledgements, path reversals or otherwise assume bi-directionality of the links.

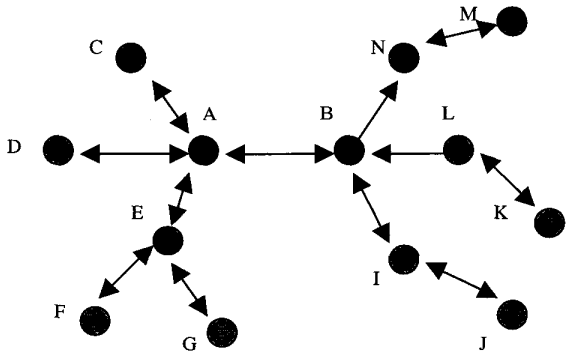


Fig. 1. Constraints on spatial label reuse

In our distributed algorithm we assign labels only to bi-directional links. However, the assignment must be correct even in the presence of unidirectional links. Also, we perform the assignment for undirected links of the underlying graph, i.e., the same label is used for traffic flowing in either direction along the link. We describe below the constraints for label assignment.

The goal is to assign a label for the link *A-B*. The idea is that each node must be able to figure out whether or not it is the intended receiver of a particular transmission. Since we restrict data communication to bi-directional links we never assign a label to links *B-N* and *B-L*. Clearly the label used for link *A-B* has to be distinct from those used for links *A-C*, *A-D* and *A-E* so that *A* can distinguish between bi-neighbors *B*, *C*, *D* and *E*, respectively. Likewise it should also be distinct from that used for link *B-I* for *B* to distinguish between bi-neighbors *A* and *I*.

Another constraint is that the label for *A-B* cannot be the same as that used for links *E-F* and *E-G*. This is so that *E* can distinguish between neighbors *A*, *F* and *G*. Similarly, it should also be different from that used for link *I-J* so that *I* can distinguish between neighbors *B* and *J*. Another important point to note is that although *B-N* is a unidirectional link, the label used for link *A-B* should be different from that used for link *N-M*. This is so that *N* can distinguish between transmissions from *B* and *M*. Similarly, the label used for link *L-K* should be distinct from that used by link *A-B*.

We define two links to be neighboring if they share a common node. The label assignment problem is stated as follows:

When data communication is restricted to bi-directional links, a link is said to have a valid label assignment when its label is distinct from that of its neighboring links and from the neighbors of the neighboring links, with unidirectional links never being assigned a label.

ASSIGNMENT ALGORITHM

Label Assignment

A label is assigned to a link only when there is a traffic flow along that link in order to reduce unnecessary protocol overhead. When a label is to be assigned to a link, one of the nodes initiates the assignment protocol and we refer to this node as the *active* node. We refer to the other endpoint of the link as the *passive* node. Figure 2 shows the protocol message exchanges. When a node has data to send to another node, it initiates the algorithm for label assignment, thus becoming the active node.

If it has no information about link labels in its neighborhood it broadcasts a HELLO message. The neighboring nodes that have information about link labels in their neighborhood respond with BROADCAST messages that contain link labels already assigned to their other links that are in use. In order to mitigate packet losses, this message is repeated REPEAT_VALUE times with a cycle period of 2.HALF_CYCLE. All the repeated transmissions are randomly delayed between 0 and HALF_CYCLE. This prevents repetitive collisions. Even with channel sensing, such collisions cannot be completely avoided due to the hidden terminal problem.

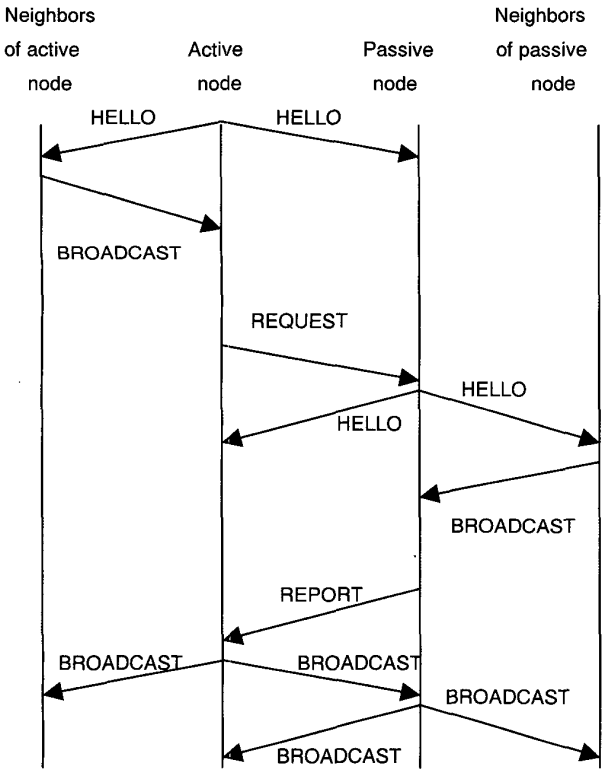


Fig. 2. Protocol message exchanges

Once the active node has obtained its neighborhood information, it sends a REQUEST message to obtain the neighborhood information of the passive node. This is needed in order to be able to correctly assign a link label in accordance with the constraints stated in section III. The reason for this is symmetry of the label that will be assigned. If the passive node does not have its neighborhood information, it obtains it by broadcasting a HELLO message. After getting its neighborhood information, it relays it to the active node by sending a REPORT message, which is repeated in the same manner as BROADCAST messages. After getting a REPORT message or a REQUEST timeout, whichever happens first, the active node selects a link label and starts broadcasting its label selection. On getting the first BROADCAST from the active node after the label assignment, the passive node also starts broadcasting.

However, there is another issue that needs to be tackled by our assignment algorithm. Consider the example in Figure 1. Suppose that link *N-M* had been assigned a label before link *A-B*. However, since node *B* cannot hear the broadcast from *N*, it will not consider the label used for *N-M* as unusable. If now link *A-B* were to be assigned the same label, it would violate the assignment constraints. The solution is that when node *N* hears the BROADCAST from node *B*, it detects this conflict. It could well be possible that node *M* also discovers this conflict through another unidirectional link. In this case both *N* and *M* would try to re-initiate the label assignment algorithm. To avoid this problem we let the node with the higher ID become the active node. Thus, if the node with the lower ID discovers a conflict, it sends a CONFLICT message to the other node. Had the conflict been discovered by the node with the higher ID, it re-initiates the protocol by sending out a REQUEST message.

The label is selected fairly quickly after the initiation of the assignment protocol. Our protocol results in an almost fixed communication overhead. Thus it is worthwhile to activate this protocol only when the energy savings that result from sending and receiving data packets outweigh the overhead. In section V, through simulations, we will derive the conditions for our protocol to result in net energy savings.

Label Representation and Selection

In the above description of our distributed algorithm we have explained how through the discovery and exchange of neighborhood information, labels are selected. Although any of the non-constrained labels can be picked, we propose for now that the smallest available label is picked first. As a direct consequence, lower labels are selected statistically more often than the higher ones. This is confirmed by the curve labeled 'lowest first' in figure 3, which shows the frequency of each selected label.

This result was obtained through simulations written using the PARSEC simulator, an event-driven parallel simulation language [9], where 1000 nodes are distributed

uniformly randomly over a field of 355 m x 355 m with a transmission range of 20 m. This results in each node having 10 neighbors on an average. Also, 5 independent traffic flows, which could have nodes in common, each with path length of 20 hops on an average, were simulated. For each flow, source and destination are on opposite sides of the field, where flows are chosen in a crossing pattern. In the next section we describe our simulation results in greater detail.

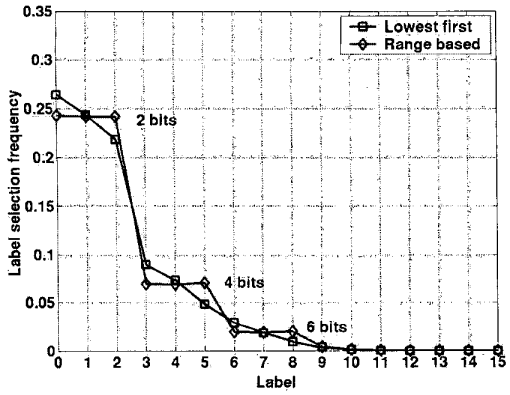


Fig. 3. Example of label selection frequency

We can exploit the non-uniform label selection distribution to our advantage and reduce the number of bits that need to be transmitted. Therefore, we propose a radical shift away from traditional MAC address representations that have a fixed number of bits for the transmitter and receiver address fields. The link label replaces both the transmitter and receiver MAC addresses. However, instead of including the actual link labels in the packet header, we propose to replace them with variable length codewords. More precisely, we use Huffman encoding, a well-known coding technique from compression theory [10].

Since we only transmit codewords, the energy efficiency of two labels with the same codeword length is the same. The lengths of the codewords depend on the relative frequencies of the labels that are chosen, which in turn depend heavily on the number of traffic streams through the network. We shall present this dependance in the next section. In practice we do not select the lowest available label, but a random one in the lowest possible range. The benefit is that it results in fewer conflicts and hence reduces the overhead caused by them. The curve in figure 3, called 'range based' illustrates the resulting label selection frequency. Every step in the staircase function corresponds to a particular range, indicated by the number of bits needed to represent the codeword.

PERFORMANCE EVALUATION

Simulation Settings

We evaluate the performance of our distributed label assignment protocol using the PARSEC simulator. In our simulations, N nodes are distributed randomly over a square field of dimensions L x L. The packet formats of our control messages are shown in figure 4.

The 'Label' field contains the codeword. The 'Neighbor info' field in BROADCAST messages is a bitmap with bits set to '1' if the corresponding link label is in use at the node that transmits this message. Similarly, the 'Constrained labels' field in the REPORT is a bitmap with bits set to '1' if the corresponding link label is constrained. The length of the bitmap field is variable with steps of 12 bits, depending on the actual labels included, where the last of the 12 bits indicates whether the bitmap ends or not. However, from the curves in figure 3 we see that labels greater 10 are almost never chosen, so the bitmap field seldom exceeds 12 bits. However, for scalability reasons, we have this provision.

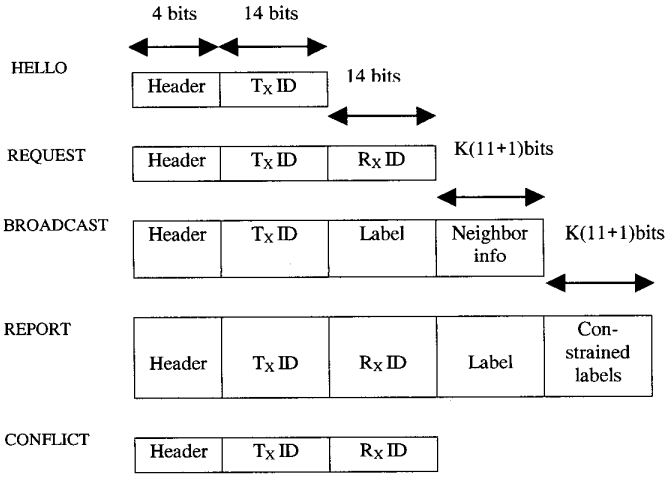


Fig. 4. Format of control messages

For our simulations we have used the parameters of the following radio and microprocessor:

- RFM radio at 2.4 kbps: 0.18 μJ/bit transmission energy and 0.04 μJ/bit receiving energy for R = 20 m, 0.94 μJ/bit transmission energy for R = 80 m [11]
- StrongARM SA-1100 processor: 1.5 nJ/instr at 150 MIPs, 2.2 nJ/instr at 250 MIPS [12]

Since we propose to transmit codewords instead of addresses, the receiver needs to perform some extra processing, as the codewords have to be stepped through bit by bit in order to detect their end. We include this cost in our total energy budget and we estimate that this operation takes 50 instructions on average. The number of nodes in the network, N, is fixed at 1000. In our performance evaluation we would like to vary the network density,

which we define as the average number of neighbors of a node, denoted by parameter λ.

$$\lambda = \frac{N}{L^2} \cdot \pi R^2$$

Another parameter that we have varied is the number of traffic streams. Since the label assignment is traffic driven, labels that are assigned depend on the network connectivity as well as the underlying traffic. For the settings that gave rise to the curves of figure 3, λ was equal to 10 and there were 5 independent traffic streams. The parameters HALF_PERIOD and REPEAT_VALUE, mentioned in section IV, are chosen as 0.2 s and 2, respectively.

Analysis

First we analyze the situation when there is only one traffic stream. In this case, a valid label assignment is as follows:

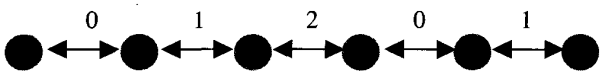


Fig. 5. Valid label assignment for single traffic stream

Thus, each label can be represented by only 2 bits. However, when there are multiple streams with nodes in common, more labels would need to be used in order to satisfy the assignment constraints. This means that the codeword corresponding to the label requires more bits. Figure 6 shows the variation of the average label size with the number of traffic streams for different node densities. We see the dependence on both the network connectivity and the traffic patterns. Note that average label size does not increase rapidly with either the number of traffic streams or the network connectivity.

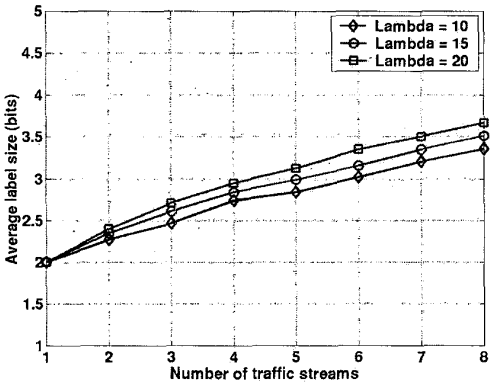


Fig. 6. Label size Vs. number of traffic streams

This demonstrates the scalability of our encoded label representation. For λ equal to 10 and 5 independent traffic streams, the average label size turns out to 2.84 bits. This small label now replaces the transmitter and receiver MAC addresses in the MAC header. Comparing this to a situation in which we use 14 bit MAC addresses we see that the address overhead is reduced from 28 bits to 2.84 bits!

Overhead vs. Energy Savings

Finally, we need to verify whether our approach does indeed result in energy savings and under what conditions. Our protocol has a certain overhead depending on the network connectivity and the number of traffic streams. For the calculation of overhead, we only consider those nodes that participate in data communication. We call refer to these nodes as *participating nodes*. This is because nodes that do not participate in data communication in their entire lifetime are anyway less important as they neither generate nor forward sensor data. These nodes that do not participate in data communication might at best be relays for network management traffic. In the set of participating nodes, we include sources and destinations of data as well as those nodes that route data packets from the source to the destination. Moreover, nodes that act as routers consume more energy than source and destination nodes because they have to send as well as receive packets. The majority of the participating nodes are router nodes.

In figure 7 we plot the energy savings per participating node as a function of the number of data packets sent by that node. The savings are with respect to the benchmark of a network that uses a 14 bit unique ID as a MAC address. We have examined the case for a network of 1000 nodes with λ equal to 10. There is one traffic stream of average length equal to 20 hops. The number of data packets sent by the node is equal to the total number of packets in that traffic stream. We compare our results with the work on assigning short, reusable MAC addresses to nodes [7].

For these settings, the average label size is 2 bits and the saving per packet is 26 bits. Recall that besides the communication overhead of our protocol, there is also the computational overhead of decoding the codewords. For the same network settings, in [7], each node was assigned a MAC address of average size equal to 4.5 bits. The saving per data packet was 2 x (14 - 4.5) bits. The overhead for each node was the following: 217 bits sent and 2082 bits received. Our calculations are for the StrongARM SA-1100 microprocessor version with 250 MIPS and the RFM radio with a 20 m range. Let P be the number of data packets sent.

Net energy saving

= MAC header savings - computational overhead - protocol overhead for bits sent - protocol overhead for bits received

$$\Delta E_{node} = 2(14 - 4.5)x(0.18 + 0.04)P - Px50x2.2x10^{-3} - 217x0.18 - 2082x0.04$$

$$\Delta E_{link} = (28 - 2)x(0.18 + 0.04)P - Px50x2.2x10^{-3} - 383x0.18 - 746x0.04$$

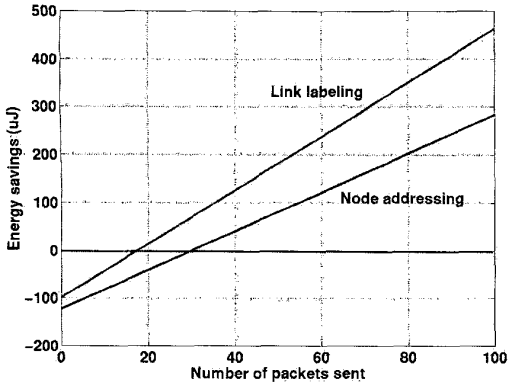


Fig. 7. Energy Savings

where ΔE_{node} and ΔE_{link} are the net energy savings per node (in μJ) for the cases of node address assignment and link label assignment, respectively.

The plots for ΔE_{node} and ΔE_{link} are labeled as 'Node addressing' and 'Link labeling', respectively. We see that our scheme for assigning labels to links outperforms our scheme for assigning addresses to nodes. It is clear that for short-lived flows there is an energy penalty. However, after a few tens of packets are sent there is a net energy saving. This is reasonable since common sensing applications such as continuous temperature monitoring, target tracking, etc. result in long-lasting traffic streams. When a large number of packets are sent, the energy savings are much more pronounced. This results in an increase in the network lifetime.

CONCLUSION

In sensor networks, the dominant traffic typically consists of packets with a small payload and a short destination name attribute. In this case, MAC addresses contribute a considerable packet header overhead. However, they cannot be simply omitted, as their functionality is needed to discern intended from non-intended receivers. In this paper, we have tackled the problem of limiting this MAC address overhead, thereby reducing the number of bits that need to be transmitted. We replace conventional MAC addresses by short link labels that are assigned dynamically. This results in a considerable reduction of the MAC header overhead. As every bit that is saved, relaxes the demands on the node's energy resources, this scheme eventually targets increasing the network operation lifetime. We have shown the conditions under which there is significant reduction in energy consumption.

ACKNOWLEDGMENTS

This paper is based on research funded in part by the ONR Minuteman Project and DARPA SensIT under AFRL contract # F30602-99-1-0529. The views expressed in the paper are the authors' and do not necessarily represent those of the funding agencies.

REFERENCES

[1] Sohrabi, K., Gao, J., Ailawadhi, V., Pottie, G., "Protocols for self-organization of a wireless sensor network," *IEEE Personal Communications Magazine*, Vol.7, No.5, pp. 16-27, Oct. 2000.

[2] Estrin, D., Govindan, R., "Next century challenges: scalable coordination in sensor networks," *Proceedings of ACM/IEEE MobiCom'99, Seattle*, WA, pp. 263-270, August 1999.

[3] Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., Lilly, J., "The design and implementation of an intentional naming system," *Operating Systems Review*, Vol.33, No.5, pp.186-201, December 1999.

[4] Bharghavan, V., "A dynamic addressing scheme for wireless media access," *Proceedings of IEEE ICC '95*, Seattle, WA, pp. 756-760, June 1995.

[5] Ramanathan, S., "A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks," *Proceedings of IEEE INFOCOM '97*, Kobe, Japan, pp. 900-907, April 1997.

[6] Giovanardi, A., Mazzini, G., Rossi, M., Zorzi, M., "Improved Header Compression for TCP/IP Over Wireless Links," *Electronic Letters*, Vol. 36, No. 23, November 2000.

[7] C. Schurgers, G. Kulkarni and M.B. Srivastava, "Distributed Assignment of Encoded MAC Addresses for Wireless Sensor Networks", *Proceedings of MobiHoc 2001*, Long Beach, CA, October 2001.

[8] Elson, J., Estrin, D., "Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks," *Proceedings of ICDCS'01*, Phoenix, AZ, April 2001.

[9] Bagrodia, R., Meyer, R., Takai, M., Chan, Y.A., Zeng, X., Marting, "Parsec: A Parallel Simulation Environment for Complex systems," *Computer*, Vol.31, No.10, pp. 77-85, October 1998.

[10] Cover, T., Thomas, J., "Elements of Information Theory," Wiley Series in Telecommunications, 1991.

[11] "Intel® StrongARM SA-1100 Microprocessor for Portable Applications Data Sheet," http//www.arm.com.

[12] "AVR 8-bit RISC - Data Sheets," http://www.atmel.com.

Appendix M:

A Framework for Efficient and Programmable Sensor Networks

Athanassios Boulis and Mani B. Srivastava

Networked and Embedded Systems Lab, EE Department, University of California at Los Angeles
email: { boulis, mbs }@ee.ucla.edu

***Abstract* – Ad hoc wireless networks of deeply embedded devices such as micro-sensors and micro-actuators have emerged as one of the key growth areas for wireless networking and computing technologies. So far these networks/systems have been designed with static and custom architectures for specific tasks, thus providing inflexible operation and interaction capabilities. Various architectures are currently trying to make sensor networks programmable and open to transient users. Most of these schemes though, promote algorithms that are too centralized and/or too interactive (i.e. the user is involved in the control loop most of the time), losing the efficiency these highly resource-limited systems need. Our approach employs active networking concepts in the form of lightweight and mobile control scripts that allow the computation, communication, and sensing resources at the sensor nodes to be efficiently harnessed in an application-specific fashion. The replication/migration of such scripts in several sensor nodes allows the dynamic deployment of distributed algorithms into the network. Although these mobile control scripts have similarities to mobile agents for traditional data networks, a framework to support them has different considerations than its traditional data network counterpart. The paper discusses these considerations and design choices, and describes SensorWare, our implementation of such a framework.[1]**

I. INTRODUCTION

In recent years we have witnessed an enormous growth in wireless networking with applications to traditional mobile computing and communication tasks. However, it is widely projected [16][32][39] that one of the highest growth applications of the wireless technology will be in the networking of deeply embedded devices such as sensors and actuators used for interacting with the physical world. In the not too distant future one will have an ultra-low power system-on-a-chip that integrates radio communication, digital computing, and MEMS sensing and actuation components on a single die. Large networks of such wireless devices may be used for applications such as premise security and surveillance, environmental habitat monitoring, condition-based maintenance, etc.

Figure 1 shows an example of a distributed wireless sensor network where an ad hoc network of miniature, resource-limited, static, wireless, sensor nodes is being used to monitor a dynamic physical environment. The use of low power communication and the need for diversity in sensing necessitates a multi-hop, distributed architecture [32]. The computation capabilities at the nodes can be leveraged for event detection via data fusion and collaborative signal processing among nearby nodes, so that higher bandwidth raw sensor data does not need to be sent to the users. Typically a user queries the network (consider the term "query" in the broad sense, not just database query), the query triggers some reaction from the network, and as the result of this reaction the user receives the information needed. The reaction to the query can vary from a simple return of a sensor value, to a complex unfolding of a distributed algorithm that promotes collaborative processing among some sensor nodes.

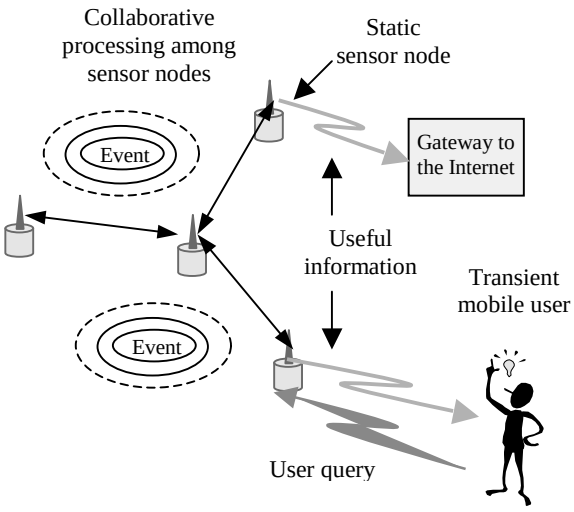


Figure 1: Distributed sensor network

These systems are also quite different from traditional networks. First, they have severe energy, computation, storage, and bandwidth constraints. Second, their overall usage scenario and the implications that this brings to the traffic and the

[1] This work was partially supported by DARPA SensIT program and ONR Minuteman project.

interaction with the users is quite different from traditional networks. These differences are discussed more extensively in section II.

Essentially Sensor Networks (SNs) are application-specific distributed systems that require a different distributed algorithm as an efficient solution to each different application problem. Given the nature of SNs, there are two classes of problems in their design. First, the application-specific problem: How does one find the most efficient distributed algorithm for a particular problem? Second, the generic problem: How does one dynamically deploy different algorithms into the network, what is the programming model that will implement these algorithms, and what general support does one need from the framework?

For the first class of problems (i.e., finding efficient algorithms for particular applications), there are many research efforts in a variety of application problems (e.g., target tracking, sensor reading aggregation). In this paper we will not expand into any particular application problem. We only note, that in general, localized distributed algorithms (i.e., distributed algorithms that act locally, using only local information) are particularly efficient in most SN problems as they achieve small and well-distributed energy consumption, thus prolonging the network lifetime.

The second class of problems (i.e., what is the right framework to express and dynamically deploy distributed algorithms for SN) is the focus in this paper. We describe our proposal of such a framework, called SensorWare. SensorWare provides a language model powerful enough to express the most efficient distributed algorithms while at the same time hiding unnecessary low-level details from the application programmer and providing a way to share the resources of a node among applications. The language model is developed after examining what are the properties of efficient algorithms for SN (e.g., localized distributed algorithms), and in conjunction with developing our own applications on real sensor networks [3].

Equally important is the role of SensorWare in the dynamic deployment of the distributed algorithms into the network. As sensor nodes are memory-constrained, they cannot store every possible application in their local memory. Thus, a way of dynamically deploying a new application is needed. Usually this means that a distributed algorithm has to be incorporated in several sensor nodes, which in turn means that these sensor nodes have to be dynamically programmed. A user-friendly and energy-efficient way of programming the nodes keeps the user out-of-the-loop most of the time by allowing sensor nodes to program their peers. By doing so, the user does not have to worry about the specifics of the distributed algorithm (because the information on how the algorithm unfolds lies within the algorithm), and the nodes save communication energy (because they interact with their immediate neighbors and not with the user node through multi-hop routes). The programming model of SensorWare is designed in such a way, so as to facilitate the user-friendly and energy-efficient dynamic deployment of an algorithm. The user "injects" the query/program into the network, and the query *autonomously* unfolds the distributed algorithm into the nodes that should be affected. The process resembles the operation of multiple collaborating mobile agents, replicating/migrating to the nodes where the distributed algorithm should be executed.

Although the resemblance with mobile agents (MA) is strong, a platform to support this kind of behavior for SN does not have the same considerations as a platform for traditional-network mobile agents. We examine these differences in section III. It is also interesting to note that if one sets aside these differences and views SensorWare as another MA system which just operates in a different realm (i.e., the SN realm), some of MA's general problems are considered solved. Kotz in [24] lists the lack of killer applications, security, performance and scalability as the most commonly pointed problems of agent technology. In SNs, most of the interesting and complex applications (e.g., target tracking) become the killer application for agent technology, exactly because they offer improved performance and scalability over other solutions. In order to achieve this, the MA paradigm should be disassociated from the notion of a single agent migrating from node to node while performing a given task, and associated with multiple simple light-weight agents that tightly collaborate to implement a distributed algorithm, while their behavior and position is influenced by physical events as well as the user needs.

Section II discusses in depth the nature of SNs, the problems of traditional static SN designs, and the general idea of our approach. Section III presents SensorWare's architecture, and discusses design choices. Section IV presents related work. Finally, section V concludes the paper.

II. MOTIVATION AND BACKGROUND

A. Diferrences of SNs with traditional data networks

The first difference of sensor networks compared to traditional data networks is that they have severe energy, computation, storage, and bandwidth constraints. For example, the wireless sensor node currently used in our implementation efforts [35] has a 133 MHz, 32-bit, Intel StrongARM 1100 CPU, 1 MB of FLASH memory, 1 MB of RAM, a 100 kbps radio, and has to operate on two 9V batteries. This is considered to be towards the high end of sensor network devices. A popular, low-end node design from UC Berkeley [21] called "mote" uses a 4MHz 8-bit Atmel CPU with 8KB of FLASH memory, only 512B of RAM, and a 10Kbps RFM radio. The major resource problem in such networks is energy, since these are static unattended networks, and the nodes cannot have renewable energy sources. Energy is so important that algorithms designed for sensor networks often sacrifice response latency, accuracy, and other user-desired qualities to save energy and prolong the operational lifetime of the network. Even with advances in battery life, energy will remain a scarce resource in the future and must be conserved and protected by the SN.

The second difference of sensor networks compared to traditional data networks is their overall usage scenario and the implications that this brings to the traffic and the interaction with the users. Typically, in traditional networks, users are connected to a node (or group of nodes) and require a service from another node. This two-entity communication model describes the overwhelming majority of traditional network traffic. The network acts as a medium bringing the two parties together. The interaction model is also straightforward; the user interacts directly with the user or the service at the other end. Certain actions from the user will produce certain data transfers to and from the other end. The most popular exceptions to these rules are free roaming mobile agents providing either data mining or broker services. However, this is a small portion of today's data networks. Sensor networks on the other hand, are less networks (i.e., in the sense that they loosely connect independent entities) and more **distributed systems**. As stated earlier, the nodes tightly collaborate to produce information–rich results. The user will rarely be interested in the readings of one or two specific nodes. The user will be interested in some parameters of a dynamic physical process. To efficiently achieve this, the nodes have to form an application-specific distributed system to provide the user with the answer. This is a departure from the two-entity model: There is not a clear second group of nodes. There are only the users and the *whole network*. The nodes that are involved in the process of providing the user with information are constantly changing as the physical phenomenon is changing.

The last remark brings us to the user-interaction topic. Apart from the user input, the physical phenomena now play a central role in the actions inside the network. The actions in each individual node are affected from external physical stimuli, information from other nodes, as well as direct input from the user. Actually, it is desirable to operate in a fashion where a node's actions are affected largely by physical stimuli detected by the node itself or nearby nodes. Frequent long trips to the user are undesirable because they are time and energy consuming. This decentralized (i.e. not all traffic flows to/from user), autonomous (i.e., user out-of-the-loop most of the time) way of operating, is called "proactive computing" (as opposed to interactive) by David Tennenhouse [39]. We also adopt the term "proactive" throughout the paper to denote an autonomous and non-interactive nature. In order for sensor networks to realize their full potential and efficiently use their limited resources, they have to be viewed as distributed proactive systems.

B. Problems of traditionally designed SNs

The computation and communication tasks to be defined in such a distributed and proactive system are very application-specific so that the system design and implementation do not lead to easy layering and abstraction of lower level details. Basically, a different distributed algorithm is needed, to provide the optimal solution for each different task. This characteristic has led to systems designed with static and custom architectures for specific tasks (e.g., [21]). It is not possible for a transient external user, even if he were to be able to interface with the network, to dynamically utilize the system resources in any other fashion than what is hardwired into the custom application. These are systems that are essentially closed to external users and systems that wish to interface with them in a way other than the pre-programmed one. The inflexible modes of operation and interaction are a hindrance when one considers that these systems usually have long deployment cycles and often face transient users with varying needs. Ideally one would want to view the

sensor network as an entity that provides services to transient users with different dynamic needs.

The notion of bringing programmability to the sensor network has been addressed before. One of the approaches currently under investigation is a distributed database model. A good example of this approach is the work done at Cornell [2]. A similar scheme called DataSpace focusing on location addressing has also been developed in Rutgers [22]. Each node is equipped with a fixed database query resolver. As queries arrive to a node, the local resolver decides on the best, distributed plan to execute the query and distributes the query to the appropriate nodes. Although this approach takes into account the distributed nature of the system and works well in several scenarios, it does not take into account the proactive nature of the system. The user is the central place of control and most data flows to/from the user. This property can prove inefficient in applications such as target tracking, where it is better for nodes to form clusters around the target, collaboratively compute the target's location and just send the location information back to the user. Clearly a more flexible way of programming the sensor network is needed to enable this kind of behavior.

C. Our approach

To address the problem of operation and interaction inflexibility, *efficiently*, taking into account both, the distributed and proactive nature of sensor networks, we have developed a framework for wireless sensor networks called ***SensorWare***. SensorWare employs active networking concepts in the form of *lightweight* and *mobile control scripts* that allow the computation, communication, and sensing resources at the sensor nodes to be efficiently harnessed in an application-specific fashion.

The SensorWare architecture is based on a scriptable lightweight run-time environment, optimized for sensor nodes that have limited energy and memory. This environment securely hosts one or more simple, compact, and platform-independent sensor-node control scripts. The sensing, communication, and signal-processing resources of a node are exposed to the control scripts that orchestrate the dataflow to assemble custom protocol and signal processing stacks. SensorWare has to also promote the creation of distributed proactive algorithms based on the scripting language described above. For this reason the scripts are made mobile using special language commands and directives. A script can replicate or migrate its code and data to other nodes, directly affecting their behavior. The replication or migration of a script will be called "population" in the paper.

A usage scenario can be like the following: A user sends a query to the sensor network. The query is a script, a state machine in its simplest form, which is injected to one or more sensor nodes. The script will describe among other things how it is going to populate itself to other nodes. The process of population can continue depending on events and the current state. For example as the events of interest are moving to a different area, the scripts can move along with them, possibly trying to predict their next move. The populated scripts will collaborate among themselves in order to extract the information needed by the user, and when this information is acquired it is sent back to the user.

Basically, SensorWare's advantage is that it allows user interaction that goes beyond simple queries that check current or past status of a node, or install triggers for event notification. In SensorWare, a query can take the form of a script, which can populate to many nodes, allowing application-specific handlers to run at these sensor nodes and process sensor events cooperatively with neighboring nodes without intervention from the user. In other words, SensorWare adopts and supports the distributed proactive model for sensor networks where each problem is handled by a different general distributed algorithm, giving the possibility for the most resource-efficient and user-optimized solution.

III. ARCHITECTURE

First, we show SensorWare's place inside the overall sensor node's architecture (Figure 2). The architecture of a sensor node can be viewed in layers. The lower layers are the raw hardware and the hardware abstraction layer (i.e., the device drivers). A real time operating system (RTOS) is on top of the lower layers. The RTOS provides all the standard functions and services of a multi-threaded environment that are needed by the layers above it. The SensorWare layer for instance, uses those functions and services offered by the RTOS to provide the run-time environment for the control scripts. The control scripts rely completely on the SensorWare layer while populating around the network. Static applications and services coexist with mobile scripts. They can use some of the functionality of SensorWare as well as standard

functions and services of the RTOS. These applications can be solutions to generic sensor node problems (e.g., location discovery), and can be distributed but not mobile. They will be part of the node's firmware.

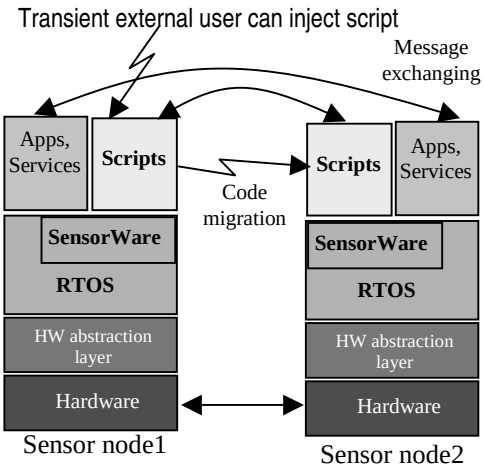


Figure 2: The general sensor node architecture

Two things comprise SensorWare: 1) the language, and 2) the supporting run-time environment. The next two subsections describe each of the parts in detail. A third subsection tries to provide a deeper insight into SensorWare by discussing some of the most important design choices.

A. The language

As discussed earlier, the basic idea is to make the nodes programmable through mobile control scripts. The choice of a script language instead of a more general model will be discussed in sub-section C.1 Here the basic parts that comprise the language will be described as well as the programming model that emerges from the parts.

First, a scripting language needs proper functions/commands to be defined and implemented in order to use them as building blocks (i.e., these will be the basic commands of the scripts). Each of these commands will abstract a specific task of the sensor node, such as communication with other nodes, or acquisition of sensing data. These commands can also introduce needed functionality like moving a script to another node or filtering the sensing data through a filter implemented in native code. Second, a scripting language needs constructs in order to tie these building blocks together in control scripts. Some examples include: constructs for flow control, like loops and conditional statements, constructs for variable handling and constructs for expression evaluation. We call all these constructs the "glue core" of the language, as they combine several of the basic building blocks to make actual control scripts.

Figure 3 illustrates the different parts of the SensorWare language. Several of the basic commands/functions are grouped in theme-related APIs. We use the term API in a generic fashion, to denote a collection of theme-related functions that provide a programming interface to a resource or a service. There are some commands that do not belong to any of the APIs and are included in the box named "other support cmds". Finally there is an important command which we chose to highlight, called "`wait`". The `wait` command defines to some extent the programming style for the scripts.

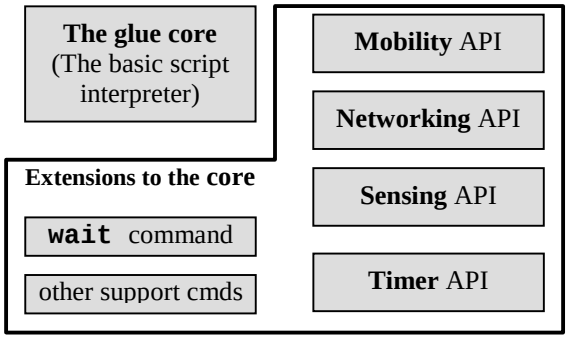


Figure 3: The language parts in SensorWare

As a glue core we can use the core from one of the scripting languages that are freely available, so we are not burdened with the task of building and verifying a core. One such scripting language, that is well suited for SensorWare's purposes, is Tcl [29], offering great modularity and portability. Thus, the Tcl core is used as the glue core in the SensorWare language. All the basic commands, such as `wait`, or the ones included in the APIs, are defined as new Tcl commands using the standard method that Tcl provides for that purpose.

The set of APIs is basically a way of easily exporting services and shared resources to the scripts. The Networking API provides the basic functions so that the scripts can communicate with each other. The Sensing API provides the basic functions for accessing and sharing sensing data from the sensors. The Timer API defines and sets/resets real time timers. Finally, the Mobility API provides the basic functions to the scripts so they can transfer themselves around the network, access the data that they carry with them from node to node and also access some very restricted local memory in the

particular node that they are currently residing. More information on each API can be found in [4].

A.1 The general programing model

As discussed earlier, according to the proactive distributed model the scripts will look mostly like state machines that are influenced by external events. Such events include network messages from peers, sensing data, and expiration of timers. The programming model that is adopted is equivalent to the following: An event is described, and it is tied with the definition of an event handler. The event handler, according to the current state, will do some (light) processing and possibly create some new events or/and alter the current state.

The behavior described above is achieved through the `wait` command. Using this command, the programmer can define all the events that the script is waiting upon, at a given time. Examples of events that a script can wait upon are: i) reception of a message of a given format, ii) traversal of a threshold for a given sensing device reading, iii) filling of a buffer with sensing data of a given sampling rate, iv) expiration of several timers. When *one* of the events declared in the `wait` command occurs, the command terminates, returning the event that caused the termination. The code after the `wait` command processes the return value and invokes the code that implements the proper event handler. After the execution of the event handler, the script moves to a new `wait` command, or more usually it loops around and waits for events from the same `wait` command.

Figure 4 shows an example of a SensorWare script. SensorWare commands are in boldface. There are comments in italics but basic Tcl knowledge is needed to follow the script. The example is sufficient to illustrate the basic programming style and the use of some of the most important commands. Without solving any real-world problem (as it randomly interacts with the physical world), the basic structure is evident: At first the script's state is accessed and updated. After setting new timers it enters the main wait loop. Events are received and processed according to their type. Scripts can also wait for a limited type of events (e.g. the second wait command of the example code). If certain conditions are met the script will replicate itself in proper neighboring nodes. The replicate command will initially notify the remote node about the code it wants to transfer. The remote node will reply on whether this code is in his cache and how many instances of the code are currently running in it, so the script can take further action. For more complicated scripts that exhibit distributed coordination and solve real-world problems we refer the interested reader to [6].

```
# These are data that the script carries with it. The particular ones inform
#the current instance about its parent node (the one which sent the script)
set send_node [Agent_memory_read 0];
set send_node_neighbors [Agent_memory_read 1];
# update these data for use in the next replication
Agent_memory_write 0 [getNodeID];
Agent_memory_write 1 [getNodeNeighbors];
#based on the above info find nodes that probably do not have the script
#and store their list in the variable $remaining nodes (commands omitted)
# set a timer called RT with the initial value 200ms
setTimer RT 200
# the big loop starts
while {1} {
      set ans [wait -msg * -data * -until RT]
#wait cmd returned, find out type and body of event.
      set type [lindex $ans 0];   set body [lrange $ans 1 end]
      switch $type {
            w {   # a timer expired, do something.
                  setTimer TT  100
            }
            s {   # data was sensed.
#wait for sensing data threshold to be passed, within 5 ms
                  set ans [wait -data -threshold 10 -until 5]
                  set type [lindex $ans 0];
                  if { $type = "s" } { set ready 1 }
            }
            n {   # a network message was received.
                  If { $ready}  {Agent_replicate $remaining_nodes; exit; }
            }
}
```

Figure 4: An example of a SensorWare script

B. The run-time environment

As important are the scripts in the SensorWare platform, equally important is the run-time environment that supports them. Figure 5 illustrates the basic tasks performed by the environment.

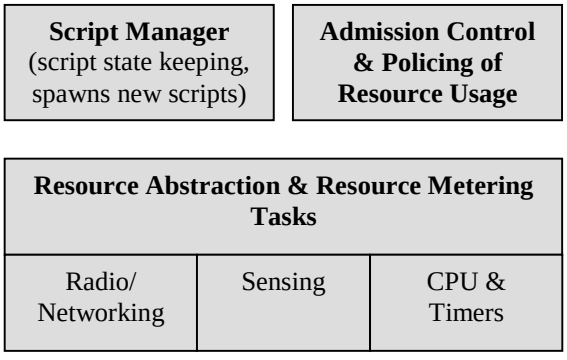


Figure 5: Tasks in the SensorWare run-time environment

The Script Manager is the task that accepts all requests for the spawning of new scripts. It forwards the request to the Admission Control task and upon receiving a positive reply, it initiates a new thread/task running a script interpreter for the new script. The Script Manager also keeps any script-related state such as script-data for as long as the script is active. Possible attacks such as snooping or spoofing are banned by the strict security model (see section C.3). The script manager also keeps a script-code cache in order to reduce code transmissions over the wireless channel. The Admission Control and Policing of Resource Usage task, as the name reveals, takes all the script admission decisions, makes sure that the scripts stay under their resource contract, and most importantly checks the overall energy consumption. If the overall consumption exhibits alarming characteristics (e.g., the current rate cannot support all scripts to completion) the task selectively terminates some scripts according to certain SensorWare policies. Resource management is discussed further in sub-section C.4 .

The run-time environment also includes three "Resource Abstraction and Resource Metering" tasks (sometimes referred to as "Resources Handling" tasks for brevity). Each task supports the commands of the corresponding APIs and manages a specific resource. For instance, the “Networking” task manages the radio: i) it implements an energy efficient routing protocol, ii) it accepts requests from the scripts about the format of network messages that they expect iii) it accepts all network messages and dispenses them to the appropriate scripts according to their needs, and finally iv) measures the radio utilization for each script, a quantity that is needed by the “Admission Control & Policing of Resource Usage” task. As another example, the “Sensor Abstraction” task manages the sensing device. It accepts all requests for sensor data from all the scripts and decides on the optimal way to control the sensing device. It also measures the sensing device utilization for each script. Finally, the "CPU and Timers" task accepts the various requests for timers by all the scripts and manages to service them using the small number (usually one or two) of real-time timers the embedded system provides. In essence the task provides many virtual timers relying on few real timers provided by the system. The task also meters the CPU utilization by each script, and can put the CPU in idle mode, if the embedded system allows this functionality.

Figure 6 depicts an abstracted view of SensorWare's run-time environment.

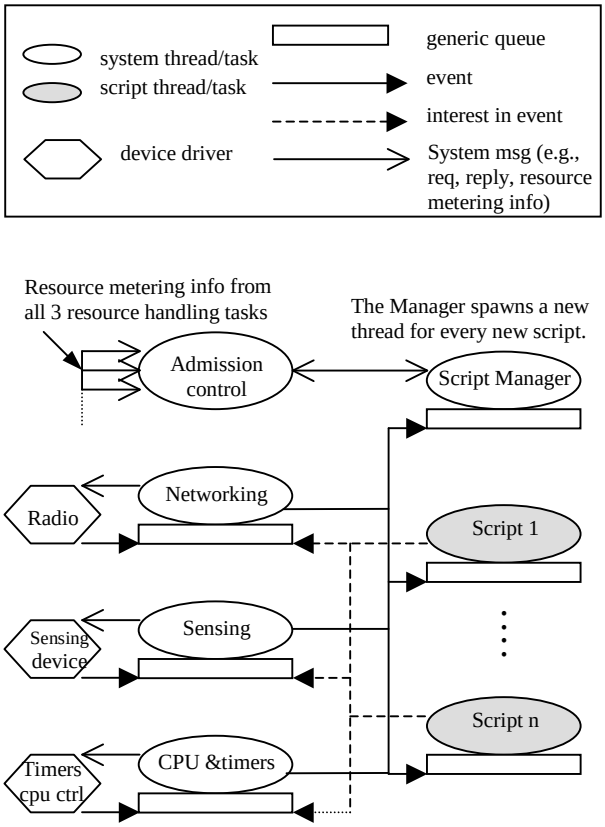


Figure 6: Abstracted view of SensorWare's run-time environment

Most of the threads running are coupled with a generic queue. Each thread "pends" on its corresponding queue, until it receives a message in the queue. When a message arrives it is promptly processed. Then the next message will be fetched, or if the queue is empty, the thread "pends" again on the queue. A queue associated with a script thread is receiving events (i.e., reception of network messages, sensing data, or expiration of timers). A queue associated with one of the three resource handling tasks, receives events of one type (from the specific device driver that is connected to), as well as messages that declare interest in this event type. For instance, the Sensing resource-handling task is receiving sensing data from the device driver and interests on sensing data from the scripts. The Script Manager queue receives messages from the network that wish to spawn a new script. There are also system messages that are exchanged between the system threads (like the ones that provide the Admission Control thread with resource metering information, or the ones that control the device drivers).

Portability is a major issue in SensorWare as we envision our system to be used by many diverse

platforms. The structure of the run-time environment remains the same across platforms, as shown in Figure 6. The only parts that need to be changed are some RTOS definitions (e.g., thread creation, queue definition and handling) for different RTOS, and interactions with the device drivers for different embedded systems (systems with different hardware and/or different design of device drivers).

C. Design choices in SensorWare

In this subsection we are going to present the issues that are important for SensorWare highlighting the differences with traditional Mobile Agents when applicable. The discussion will concentrate on: 1) The scripting abstraction 2) Coordination and mobility models, 3) Code safety and security, and 4) Resource management.

C.1 Scripting abstraction

The SensorWare scripting model has its philosophical roots in various successful systems where an embedded run-time scripting environment provides users with scriptable access to the resources of a complex system, such as JavaScript in web browsers and servers, and Tcl in simulators (e.g. ns), CAD tools (e.g. Synopsys tool chain), and RTOSs (e.g. VxWorks).

The expressiveness of our programming model does not impair even to a minimum the possible distributed algorithms developed. On the contrary, programmers of distributed proactive algorithms for SNs find it natural to program in the SensorWare language. This is because the programmers do not need lower-level access to the nodes but rather a simple glue core and a set of well-defined and functionality-rich APIs. The choice of a scripting language provides SensorWare with many advantages. First, in scripting languages it is much easier to provide a sandbox execution environment [40], thus helping code safety, a critical need when mobile code is considered. Second, the interpreted version of the scripts makes them easily portable, an important advantage when multiple sensor node platforms are considered. Third, the particular Tcl core is so lightweight and easily customizable that it is ideal for the restricted environment of a sensor node.

C.2 Coordination and mobility models

It is important to begin the discussion in this area by stating the major difference between mobile agents for traditional data networks and SensorWare's mobile scripts. Stated succinctly, their intended uses and deployment scenarios are different. A traditional mobile agent is meant for an Internet-application environment possibly with mobile ends and intermittent connections. It is viewed as a free-roaming entity, mostly autonomous and self-contained (i.e. the application usually consists of one agent; there is no need for distributed computing). The agent executes in a machine until certain conditions are met, and then it migrates to the next suitable machine to execute the same or different code portions. SensorWare's scripts on the other hand are rarely an application on their own. An application will consist of many simple scripts, executing in different nodes, which are *tightly* collaborating among themselves. This set of scripts will probably expand/shrink and move as physical phenomena are evolving. The mobility of the scripts is used to properly diffuse the distributed algorithm into the network.

This major difference in intended uses and deployment scenarios is the cause for all other operational and architectural differences. One such operational difference is the coordination model. Sometimes it is desirable for traditional agents to coordinate among themselves. Because this coordination is usually done in a loose manner and the agents usually roam in networks with intermittent connections, the direct communication model of a client-server scheme often makes programming hard. Several enhanced schemes were proposed in the literature to alleviate the agents from knowing the exact address of the other agent, as well as the exact time that they wish to communicate. For example, the meeting-oriented model [42], the blackboard model [9][13], and the Linda-like model [1][10], all achieve increasingly higher spatial and temporal uncoupling of the agents. For SensorWare scripts on the other hand, the direct communication model is perfectly suited. As stated earlier the scripts are envisioned to perform very tight collaboration among them. In addition, this kind of collaboration will happen among locally clustered and static nodes, making the peer-to-peer direct communication easier. Spatial and temporal uncoupling mechanisms will just introduce an unwanted overhead. The main command used for message passing in SensorWare is send <dest> <data>, with dest::=<nodeid:name of code:user:instance>, and data::=<string> (e.g. send *:aggregation.max.temp:* "1 3.5" sends the string "1 3.5" to all scripts named "aggregation.max.temp" residing in any of the neighboring nodes, belonging to any user).

Another architectural difference of SensorWare with respect to data network frameworks is the mobility model. In most traditional mobile agent environments there is a command to capture and restore the complete state of a migrating agent. This way, the execution of an agent could be stopped at an arbitrary point in one node, and resume execution in another node from the exact same point, as nothing happened. This form of mobility is called "strong mobility" as opposed to "weak mobility" that SensorWare supports. Weak mobility basically supports the transportation of the code and data to a remote node, but the execution starts from well-known fixed entry points in the code. Weak mobility is also supported by some commercial agent systems. The prime reason for this is that strong mobility requires customized interpreters, thus making penetration in the market more difficult. SensorWare's reasons on the other hand are different. Firstly, SensorWare must be lightweight. Secondly and more importantly, there is no evident need for strong mobility in our scenario. There in no need for agents that hop from node to node, each time doing a portion of the work. SensorWare needs only an efficient way to diffuse the distributed algorithm into the network.

C.3 Code safety and security

We distinguish between code safety and security in the following sense: code safety relates to the execution of a script in the SensorWare run-time environment inside a node, whereas security relates to the network as a whole. For code safety, one would want guaranties that a buggy or malicious script will not have any effect on other scripts or on the run-time system. For security, one would want guaranties that an intruder could not gain access to resources or information of the network, and could not affect the use of the network by legitimate users. SensorWare does not consider general security issues. Current SNs though, have fewer security concerns than data networks, because they are assumed to have one user or a set of collaborating users. The major problems are authenticating the current set of users and deny any service to anyone else, as well as encrypt the data. Wen et al. [41] describe a security scheme for sensor networks that could easily work alongside with SensorWare. If the problem of legitimate access to the network is solved, code safety is the only issue in achieving overall security.

Code safety is an integral part of SensorWare as it is closely related to the language and run-time environment design choices. SensorWare employs a simple and strict code safety model. The model adopts the sandbox environment approach, which is one of the four practical techniques for mobile code safety [36]. Usually this approach is coupled with code-signing (like the Java model), in order to associate different code portions with different resource access levels. This is not needed in SensorWare though, as all scripts have the exact same privileges. The sandbox environment is not a single consideration in our framework but rather emerges from the design choices made in the language, in the way scripts are executed in the run-time environment, and in the resource management.

One needs to provide protection under the following possible attacks [12]: i) information leakage, ii) information tampering, iii) resource stealing, iv) antagonism (no gain for the attacking script, but harmful for the attacked scripts). At the language level, the choice of an interpreted scripting language provides many advantages. No low-level control is available in the scripts and the interpreter provides an ideal framework to sandbox the executing programs [40]. Furthermore, our scripting framework is stripped down of any commands that could directly access global state or global resources. The only shared resources are accessed under restrictive APIs. Basically the APIs allow the scripts to declare 'interest' in the resources and the appropriate resource-handling task is capable of safely processing multiple requests from multiple scripts. In addition, when multiple scripts are running on a node they execute in different threads under different interpreters having their own thread stack. The interpreters are altered so that they operate only within the limits of the stack. Thus it is virtually impossible for one script to affect accidentally or deliberately the state of another script or the system's state. Possible security breaches, like resource stealing, or denial-of-service attacks (antagonism), are eliminated by the resource management in SensorWare, which strictly police all scripts at run-time. The resource management issues are discussed in the following sub-section.

C.4 Resource management

Even though there are no competing users, there are competing applications; thus one cannot blindly replicate/migrate any script at any node. The resources are limited (especially energy) and must be shared among applications. Even in the case where a single application exists in the whole network, it

might choose to alter its behavior to conserve energy. In SensorWare, resources are metered at run-time, for all scripts, by the corresponding resource handling tasks, and a central task makes admission decisions and polices the scripts according to their pre-negotiated values. As mentioned previously, the premium resource in sensor networks is energy. The energy consumed by a script running on a node depends on many attributes. Generally, the script's usage profile of the node's modules (e.g., radio module, CPU, sensing module) in conjunction with the profiles of the other scripts currently sharing the node's environment will determine the overall energy consumption, and the amount that the specific script is contributing to this quantity.

Since some resources are sharable by the scripts (e.g. radio in receive mode, sensing device), while others are not sharable (e.g. the radio in transmit mode, the CPU), the task of initially estimating, and later measuring the impact of an admitted script on the energy consumption, is especially difficult. As an example, imagine that a script, already running in a node, needs to have the radio in listening mode, 100% of its lifetime. An incoming script, with the same listening requirements and zero transmitting requirements, should have little effect on the overall consumed energy and could be easily admitted. If the first script was absent though, the admission decision could be different.

In SensorWare the profile of the script consists of the values: 1) script's lifetime in a particular node, 2) percentage of lifetime the radio is in “listen” mode, 3) transmitted bytes, 4) percentage of lifetime that the sensing module is active, and 5) time that the CPU is active. These values along with a quantity named “importance indicator” that introduces a usefulness metric for the scripts, will determine the admission and further survival of the script in a node. The energy-based admission control and policing rules are an integral part of SensorWare and involve intricate issues. Some of these issues include the real-time measurement of the script’s profile values, as well as the added energy load of a script with respect to a specific set of scripts running in the node. Other issues involve alternate definitions of usefulness metrics, and analysis of heuristic admission and policing rules to maximize the useful work done, under energy constraints. Our research explores certain heuristic rules and reports 33%-86% increase in useful work done (i.e., sum of scripts completed multiplied by their importance indicator), under finite energy supplies. Resource management will not be further analyzed in this paper; more can be found in [4].

IV. Related Work

SensorWare falls under the broad family of active networking frameworks. As we mentioned its closest relatives are Mobile Agent frameworks, which we compared against SensorWare throughout the text. One might wonder how does SensorWare compares with other active networking frameworks, especially some that exhibit obvious similarities. One such framework is the PLAN language [20]. PLAN, as SensorWare, uses a scripting abstraction to describe simple packet programs based on more advanced node-resident services, adopts a simple but strict language model to achieve safety (and resource management, unlike SensorWare), and supports the weak mobility model. However, the goals of PLAN and SensorWare are different which lead to subtle but important distinctions. First and foremost PLAN and SensorWare have completely different programming models. SensorWare is event-driven, with a script interacting with the physical world and other scripts. PLAN is packet oriented, focused on executing simple programs as packets pass through the active nodes. As a “side-effect” SensorWare's APIs offer different services than PLAN's build-in functions. Less obvious are the differences in resource management. PLAN, focusing on network traffic, sets bounds on resources used (CPU, memory, bandwidth) through the design of the language. For instance, a program has predictable termination time; linearly proportional to the packet size that is carrying it. This model is just not acceptable in our case as we depend on unpredictable external physical events. Furthermore, the prime resource is energy, complicating things as discussed in section III.C.4 .

It would also be helpful to compare SensorWare with other programming platforms for SN, such as TinyOS from Berkeley [21]. TinyOS can be viewed as a node-level operating system, much like any other embedded OS, having different design focus though from traditional embedded OS. SensorWare can be viewed as a system-level OS, which is based on the existence of a node-level OS. Essentially, the two frameworks belong into different categories, with TinyOS trying to provide efficiency at the node level, and SensorWare working at the system level. Furthermore their whole philosophies are different. SensorWare wants to make SN open at runtime. TinyOS focuses on efficient programmability in the

pre-deployment phase. Finally, one might be interested to know if TinyOS can be used as a substrate for SensorWare. This cannot be done, as TinyOS architecture is so simple (by design) that does not support common features of embedded OS such as multi-threading, needed by SensorWare.

In the rest of the section we only consider work that tries to make SN programmable using active network concepts. Therefore, general mobile agent platforms are not discussed any further, nor any distributed database systems for SN are presented.

Particularly instructive is to study the relationship between SensorWare's mobile scripting approach and the mobile code approach in Penn State's Reactive Sensor Network [34] (RSN) project under DARPA's SenseIT program [37]. RSN's focus is on providing an architecture whereby sensor nodes can: (i) download executables and DLLs, identified by URLs, from repositories or their cache, (ii) execute the program at the local node using input data which itself may be remotely located and identified by a URL, and (iii) write the data to a possibly remote URL. The RSN model is in essence Java's applet model generalized to arbitrary executables and data, and combined with a lookup service. The focus of RSN is quite different from SensorWare. Differences include: (i) RSN provides a general lookup and download service, (ii) RSN does not seek to provide a scripting environment with an associated sensor node resource model for use by scripts, and (iii) RSN's notion of mobility is download oriented, as opposed to SensorWare's approach of a script which can autonomously spawn scripts to remote nodes. RSN views sensor nodes as network switches with dynamically adaptable protocols, trying to directly map the motivation and methods of classical active networks into sensor networks. Unfortunately such an approach does not address the basic problems of sensor networks. Although one might be able to construct some distributed applications using the above scheme, by no means the creation and diffusion of distributed proactive applications into the network is supported by its architecture.

Finally, extremely relevant is the work that is being conducted in University of Delaware by Jaikaeo et al. [23] called SQTL (Sensor Querying and Tasking Language). Having the same goals as our research, but starting from a different point (database-like queries), the researchers end up with the same basic solution as SensorWare, namely a tasking language for sensor networks. To lively demonstrate the relevance to our work we are quoting an excerpt from [23].*"We model a sensor network as a set of collaborating nodes that carry out querying and tasking programmed in SQTL. A frontend node injects a message, that encapsulates an SQTL program, into a sensor node and starts a diffusion computation. A sensor node may diffuse the encapsulated SQTL program to other nodes as dictated by its logic and collaboratively perform the specified querying or tasking activity."*

SQTL fits in a more general architecture for sensor networks called SINA (Sensor Information Networking Architecture) [38]. SINA uses both SQL-like queries as well as SQTL programs. Some of its main features include: 1) hierarchical clustering, 2) attribute-based naming, 3) a spreadsheet paradigm for organizing sensor data in the nodes. SQL-like queries use these three features to execute simple querying and monitoring tasks. When a more advanced operation is needed though, SQTL plays the essential role by programming (or "tasking" as the researchers from Delaware call it) the sensor nodes and allowing proactive population of the program. In SINA, SQTL is used as an enhancement of simple SQL-like queries. The framework is there mainly to support the queries not the mobile scripts. As a consequence, SQTL scripts do not have all the provisions that SensorWare scripts have. The most important of them are: 1) Rich sensor-node-related APIs (e.g. for networking, sensing). 2) Diverse rules for mobility. A SQTL script can only specify the nodes to be populated. SensorWare first checks if the script is already in the remote node and offers a multitude of possibilities depending on how many instances of the script are already running in the remote node. 3) Code modularity in order to share functionality and avoid redundant code transfers 4) Support for multi-user scripts. 5) Resource management in the presence of multiple scripts running in the node. For more information on SensorWare's provisions for efficient applications in SN one can refer to [5].

V. CONCLUSIONS

In this paper we argue that the development of a framework based on a lightweight mobile scripting mechanism will help bring many desired properties in sensor networks. It will make the sensor networks programmable and open to external users and systems, keeping at the same time the efficiency that distributed proactive algorithms have. We present the framework's architecture and discuss design choices.

SensorWare has been used in building distributed algorithms for SN problems with very promising results [3].

VI. REFERENCES

[1] S. Ahuja, N. Carriero, D. Gelernter, "Linda and Friends", *IEEE Computer*, Vol. 19, No.8, pp. 26-34, August 1986.

[2] P. Bonnet, J. Gehrke, and P. Seshadri, " Querying the Physical World", IEEE Personal Communications, October 2000.

[3] A. Boulis and M. B. Srivastava, "Aggregation applications in resource-constrained distributed systems" TM- UCLA-NESL-2001-11-002, http://nesl.ee.ucla.edu/TM/

[4] A. Boulis, "Designing Proactive Sensor Networks", Prospectus of the Ph.D. dissertation, Electrical Engineering Dept. at UCLA, March 2001, http://www.ee.ucla.edu/~boulis/phd/Qual_proposal.pdf

[5] A. Boulis and M. B. Srivastava, "Enabling Mobile and Distributed Computing in Sensor Networks", TM-UCLA-NESL-2001-07-001, http://nesl.ee.ucla.edu/TM/

[6] A. Boulis, "Illustrating Distributed Algorithms for Sensor Networks", http://www.ee.ucla.edu/~boulis/phd/Illustrations.html

[7] G. Cabri, L. Leonardi, and F. Zamponelli, "MARS: A programmable coordination architecture for Mobile agents", IEEE Internet Computing, vol. 4, no. 4, pp. 26-35, Jul.-Aug. 2000.

[8] G. Di Caro and M. Dorigo, "Mobile Agents for Adaptive Routing", Proceedings of the 31st Hawaii International Conference on System, IEEE Computer Society Press, Los Alamitos, CA, 74-83, 1998.

[9] L. Cardelli, D. Gordon, "Mobile Ambients", Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science, No. 1378, Springer-Verlag (D), pp. 140-155, 1998.

[10] P. Ciancarini, R. Tolksdorf, F. Vitali, D. Rossi, A. Knoche, "Coordinating Multi-Agents Applications on the WWW: a Reference Architecture", IEEE Transactions on SoftwareEngineering, Vol. 24, No. 8, pp. 362-375, May 1998.

[11] L. Clare, G. Pottie, J.R. Agre, "Self-Organizing Distributed Sensor Networks", Proceedings of SPIE conference on Unattended Ground Sensor Technologies and Applications, pp. 229-237, April 1999.

[12] G. Coulouris J. Dollimore, amd T. Kindberg, "Distributed systems-concepts and designs", Chapter 16, Addison-Wesley, 2nd edition, 1994.

[13] P. Domel, A. Lingnau, O. Drobnik, "Mobile Agent Interaction in Heterogeneous Environment", 1st International Workshop on Mobile Agents, Lecture Notes in Computer Science, Springer-Verlag (D), No. 1219, pp. 136-148, April 1997.

[14] M. Dorigo, V. Maniezzo & A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Agents", IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29-41, 1996.

[15] eCos: Embedded Configurable Operating System, http://sources.redhat.com/ecos/

[16] D.Estrin, R.Govindan, J.Heidemann (Editors), "Embedding the Internet", Communications of the ACM. Vol. 43, no 5, pp. 38-41, May 2000.

[17] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", ACM Mobicom Conference, Seattle, WA, August 1999.

[18] R.S. Gray, "Agent Tcl: a flexible and secure mobile-agent system", Proceedings of 4th Annual Tcl/Tk Workshop '96, Monterey, CA, USA, 10-13 July 1996. p.9-23. 235 pp.

[19] R. S. Gray, "Agent Tcl: a flexible and secure mobile-agent system", Dr. Dobb's Journal March 1997.

[20] M. Hicks, P. Kakkar, J. Moore, C. Gunter and S. Nettles, "PLAN: A Packet Language for Active Networks", Proceedings of the International Conference on Functional Programming (ICFP '98), 1998.

[21] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System Architecture Directions for Networked Sensors", Proceedings of ASPLOS-IX, November 2000 Cambridge, MA, USA

[22] T. Imielinski and S Goel, "DataSpace: Querying and monitoring deeply networked collections in physical space", IEEE Personal Communications, Oct. 2000.

[23] C. Jaikaeo, C. Srisathapornphat, and C. Shen, "Querying and Tasking of Sensor Networks", SPIE's 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Control (Digitization of the Battlespace V), Orlando, Florida, April 26-27, 2000.

[24] D. Kotz, R. Gray, "Mobile Agents and the Future of the Internet", in ACM Operating Systems Review, 33(3), 1999.

[25] J. Labrosse, " MicroC/OS-II: The Real Time Kernel", CMP Books, November 1998.

[26] D. Lange, M. Oshima, "Programming & Deploying Mobile Agents with Java Aglets", Addison-Wesley, 1998.

[27] P. Marques, P. Simoes, L. Silva, F. Boavida J. Gabriel, "Providing Applications with Mobile Agent Technology", Proceedings of the Fourth IEEE Conference on Open Architectures and Network Programming, (OPENARCH'01), Anchorage, 2001.

[28] NS-2 Simulator, http://www.isi.edu/nsnam/ns/

[29] J. K. Ousterhout, "Scripting: higher level programming for the 21st Century", Computer, vol.31, (no.3), IEEE Comput. Soc, March 1998. p.23-30.

[30] J. K. Ousterhout, "Tcl and the Tk toolkit", Addison-Wesley, 1994.

[31] S. Park, A. Savvides and M. Srivastava, "SensorSim: A Simulation Framework for Sensor Networks", Proceeding of MSWiM 2000,Boston, MA, August 11, 2000.

[32] G.J. Pottie and W.J. Kaiser, "Wireless Integrated Network Sensors", Communications of the ACM. Vol. 43, no 5. May 2000.

[33] M. Ranganathan, V. Schaal, V. Galtier, and D. Montgomery, "Mobile Streams: a middleware for reconfigurable distributed scripting", First International Symposium in Agent Systems and Applications, pp. 162-175, Oct. 1999.

[34] Reactive Sensor Networks, http://strange.arl.psu.edu/ RSN/

[35] Rockwell WINS nodes, http://wins.rsc.rockwell.com/

[36] A. Rubin, and D. E. Geer, "Mobile Code Security", IEEE Internet Computing, November-December 1998.

[37] SenseIT program, http://www.darpa.mil/ito/research/sensit/index.html

[38] C. Srisathapornphat, C. Jaikaeo, and C. Shen, "Sensor Information Networking Architecture", International Workshop on Pervasive Computing (IWPC'00), Toronto, Canada, August 21-24, 2000.

[39] D. Tennenhouse, "Proactive Computing", Communications of the ACM. Vol. 43, no 5, pp.43-50, May 2000.

[40] T. Thorn, "Programming Languages for Mobile Code" INRIA Technical Report No. 3134, March, 1997.

[41] V. Wen, A.Perig, R. Szewczyk, "SPINS: Security suite for Sensor Networks", Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, Rome, Italy, July 16-21, 2001.

[42] J. White, "Mobile Agents", in J. Bradshaw Editor.: Software Agents, AAAI Press, MenloPark (CA), pp. 437-472, 1999.

Appendix N:

The Bits and Flops of the n-hop Multilateration Primitive For Node Localization Problems

Andreas Savvides, Heemin Park and Mani B. Srivastava
{asavvide,hmpark,mbs}@ee.ucla.edu
Networked and Embedded Systems Lab, EE, UCLA

ABSTRACT

The recent advances in MEMS, embedded system and wireless communication technologies are making the realization and deployment of networked wireless microsensors a tangible task. Vital to the success of wireless microsensor networks relies is the ability of microsensors to "collectively perform sensing and computation"[1]. In this paper, we study one such challenge, that of ad-hoc node localization. The distributed computation method described here enables ad-hoc deployed wireless microsensors to discover their physical locations with an accuracy of a few centimeters. Each sensor "senses" its physical distance from its neighbors and computes an accurate estimate of its physical location. Although individual microsensors have neither the information nor the computation and memory resources to individually perform the vital task of node localization, they can collectively solve this problem in a robust, distributed manner. The key to our approach is the *n-hop multilateration primitive*, an operation that can estimate the locations of nodes that are multiple hops away from known reference points. Our simulation results show that a 3-centimeter accuracy [1] position estimates with respect to the global topology can be computed locally with linear computation cost (3 to 4 MFLOPs per node) , as opposed to the centralized computation cost which appears to be cubic with respect to the unknown nodes and without compromising the accuracy of the computed result.

1. INTRODUCTION

The marriage of ever tinier and cheaper embedded processors and wireless interfaces with micro-sensors based on micro-mechanical systems (MEMS) technology has led to the emergence of wireless sensor networks as a novel class of networked embedded systems. Many interesting and diverse applications for these systems are currently being explored. In indoor settings, sensor networks are already being used for condition-based maintenance of complex equipment in factories. In outdoor environments, these networks can monitor the natural habitats, remote ecosystems, endangered species, forest fires, and disaster sites. The ultimate goal of these networked sensors is to "coordinate amongst themselves to achieve a higher sensing task" [1]. Because of these requirements, sensor networks are expected to act as large distributed systems where computation is embedded inside the network. The nature of this problem imposes new challenges on how to coordinate and distribute computation on cheap, energy-constrained devices with limited memory and computation capabilities. UC Berkeley's Rene mote for example has a 4MHz 8-bit micro controller from Atmel with 512 bytes of FLASH memory and 8 kilobytes of RAM, a low power radio and some sensors. With such limits, it is often the case that many nodes are required to perform collaborative computation on a sensed event. In this paper we examine one such problem, that of ad-hoc node localization.

Our work is motivated by the fact that many sensor network applications are based on the ad-hoc deployment of wireless sensors in dangerous or highly toxic environments to monitor different events and report their findings back to a remote operator. In such context, it is of utmost importance that each node is aware of its location. Such knowledge of node location is needed for reporting the geographic origins of sensed events and to perform other network level functions such as geo-routing [7] and network managment. Furthermore, recent work in ad-hoc and sensor networks [9], [7] and [8] which build up on the assumption of known node locations reinforces the premise that multihop node localization is becoming an indispensable feature for applications in this new era of ubiquitous computing. For this purpose we have developed the *n-hop multilateration primitive* to estimate node locations over multiple hops. To estimate their locations, wireless devices use different technologies (i.e ultrasound, acoustics, laser, radio signal strength) to measure distances (perform ranging) to fixed landmarks. Using this ranging information and the known locations of landmarks, an optimization problem can be formulated and solved to obtain estimates of node physical locations. Besides ad-hoc deployment, the n-hop multilateration primitive would be useful in other setups of deeply embedded systems and context aware applications such as security applications and asset and personnel tracking inside buildings. In indoor infrastructure settings for instance, the n-hop multilateration primitive can assist localization in the presence of obstacles. In other situations, it can improve system robustness or simply complement existing methods to improve local-

[1] Result based on average, details in figure 15

ization accuracy. We refer to our method as a primitive because it can be used as a component in larger systems in many different configurations. The fully distributed form of the primitive on one hand, is sufficient to perform localization on a minimal system that uses a simple medium access control protocol does not require any routing protocol. The centralized computation model on the other hand, can be used as part of a greater system that is either centralized or locally centralized. In such case, the communication requirements can be easily adjusted to work in harmony with routing or any other communication and coordination protocols in the system.

1.1 What is presented in this paper

Our contribution is a method with which ad-hoc sensor nodes can coordinate among themselves to solve a global non-linear optimization problem of multihop localization, thus waiving the line of sight requirement with beacons. In this paper we present two computation models: centralized and distributed. The main goal of our discussion is to demonstrate the capability of our distributed approach to compute the global optimum locally. In the setup we investigate, the optimal position estimate for each node is the one computed from a global vantage point that considers all the available location information and all distance measurements between neighboring nodes. The proposed scheme addresses this challenge by using a very small number (3 or more) of initial landmarks (the beacon nodes) and by utilizing location and distance measurement information over multiple hops. Although none of the nodes has sufficient memory and computation resources to solve this problem individually, the nodes collectively solve this problem by performing local computation and communication. A distinct feature of our approach is the introduction of a three-phase process for estimating the locations of nodes that are physically located multiple hops away from the beacon nodes. In the first phase, nodes organize themselves in special configurations, the *computation subtrees*, which allow nodes to estimate their location over multiple hops while preventing error accumulation. During the second phase, the nodes use simple geometrical relationships to obtain a crude initial estimate of their locations. These estimates are later on refined in the third phase (position refinement phase) using a Kalman filter [18]. The third phase has two possible models of computation, centralized and distributed. The distributed computation model presents another novelty of our approach. By performing the Kalman Filter computation in the context a computation subtree, nodes in a multihop network establish a gradient with respect to the global topology constraints. This gradient enables each unknown node member of the computation subtree to estimate its globally optimal position estimate by performing computation locally. Our results demonstrate that n-hop multilateration makes it feasible for a group of small computationally constrained nodes to collectively solve the non-linear optimization problem of node localization, a problem that none of the nodes can solve individually.

This paper has two main goals 1) to describe the computation model of the n-hop multilateration primitive (centralized and distributed) and 2) to illuminate some of the inherent properties of the n-hop multilateration primitive by evaluating the different trends in computation, localization accuracy, communication and latency. The results presented here are demonstrated by a set of simulations in ns-2 and MATLAB that compare two different implementations, one centralized and one distributed. Furthermore, we developed our discussion around our experimental wireless sensing device for localization, described in section 7.4. We use this device to demonstrate the viability of our approach by implementing this algorithm on a research oriented wireless sensing device we have developed for conducting experiments related to node localization.

2. RELATED WORK

Node localization has been the topic of active research and many systems have made their appearance in the past few years. Microsoft's RADAR system [16] and UCLA's GPS-less localization system [17] provide RF based localization. In both cases the unpredictability of wireless links hindered localization accuracy and enforces these systems to rely on preplanned infrastructures. The AT&T Laboratories Active Bats [14], Intersense's Constellation [15] and MIT's Crickets [6] are all infrastructure-based localization systems based on ultrasonic distance measurements. These three systems can perform localization with a few centimeter accuracy but they are all infrastructure based. So forms of ad-hoc localization made its appearance in the domain of mobile robotics. An example of the latest work in this field is Roumeliotis's Synergetic Localization Scheme [13] in which a group of mobile robots localize themselves by combining a set of inertial measurements and distance measurements using a distributed Kalman Filter. Like the method presented here this approach uses a distributed Kalman Filter but has a different setup. The robots start from the same starting point and they keep track of each other by exchanging measurement data and location measurements. This enables the robots to localize themselves with respect to each other. Another difference is that this approach does not consider computationally constrained systems in large numbers, which have to utilize measurements and location information over multiple hops.

The AHLoS system [2] is another ad-hoc localization system. It is based on an iterative multilateration in which nodes unknown nodes that estimate their locations become beacons that in turn help other nodes to localize themselves. A problem with this approach is error accumulation in the network. Also compared to the system presented here, iterative multilateration requires more nodes to be initially configured as beacons. The authors of the AHLoS paper also mention another method, collaborative multilateration but they do not provide any details of how this operation can be efficiently performed. We show how this operation can be done with very few beacons.

Finally, Doherty's [10] convex localization approach describes a method for localizing nodes in an ad-hoc network. This method is based on semi-definite programming and requires rigorous centralized computation so it is not always suitable for low-cost wireless devices. Our approach is different from the previously proposed approaches in the sense that it can estimate locations of nodes that are found multiple hops away from the beacons while avoiding error propagation. We limit error propagation by performing node computation in the context of a computation tree. Additionally, the

proposed approach is flexible, since it can run in many different configurations: fully distributed, locally centralized at a cluster head or purely centralized.

3. MOTIVATION AND ASSUMPTIONS

In many situations, wireless sensor networks are expected to be deployed in an ad-hoc fashion (i.e air-dropped over an area). With ad-hoc deployment however, one cannot accurately predict or plan a-priori the location of each sensor. As mentioned earlier however, for many applications, sensor nodes are expected to know their physical locations. It is therefore essential to endow wireless sensor nodes with the capability to dynamically estimate their locations. The authors of the GPS-less localization system [17] argue that GPS cannot be used in all occasion due to power, form factor and the line of sight requirement. We share a similar viewpoint hence we seek to develop a system that does not require direct line of sight with beacons so that can be easily deployed in an ad-hoc manner, yet can achieve the same level of localization accuracy. To meet this goal, nodes with unknown node positions collaborate and share information with other unknown nodes over multiple hops to collectively estimate their locations. This problem has a mutual relationship to ad-hoc networking. From the localization perspective on one hand, information needs to efficiently propagate over multiple hops before the problem is solved. For wireless devices on the other hand, localization is an important feature for enabling context awareness.

3.1 Problem Statement

Given a network of sensor nodes where a) only a small fraction of the nodes is aware of their initial locations (the beacons), and b) sensor nodes within radio range of each other can also measure the distance between each other, estimate the locations of the remaining nodes (the unknowns).

3.2 Ranging Technologies

The only assumption that we make in our algorithms is that nodes can accurately measure distances between themselves and their neighbors. This assumption is supported readily available technologies, which can be found in many existing systems. The Medusa node in [2] performs some basic ultrasonic distance measurements has a 3-meter range and 2-centimeter accuracy. More advanced systems can achieve higher ranges and better resolutions. Hexamite [20] advertises ranging devices based on modulated ultrasound with a 20-meter range and 0.3 centimeter accuracy. InterSense [4] Soni discs are higher end devices that also have sub centimeter precision. All three devices use 40KHz ultrasonic transceivers and are based on time-of-arrival measurements. Medusa measures the time-of-arrival between RF and ultrasonic transmissions. In Hexamite's system the beacons transmit an ultrasonic scan that queries each of the devices. Distance is measured by recording the time taken for each device to respond. Soni Disks measure time-of-arrival by recording time between the reception infrared and ultrasonic distances. Girod's NLOS wideband acoustic ranging system can also deliver very accurate distance measurements. In outdoor environments this system has a 50-meter range. Reference [5] provides a detailed study of this system. Laser ranging technologies can make accurate distance measurements at longer ranges. The SICK's [11] laser range finders have an effective range of 80 meters and 1 centimeter accuracies.

For indoor applications of pervasive computing we are in favor of ultrasonic distance measurements because it is silent, accurate, low cost and relatively low power. In our distance measurement experiments we have experimented two families of with 40KHz ultrasonic transducers, ceramic and polymer. Ceramic transducers have greater degree of directivity while ceramics can transmit in wider angles. Ceramic transducers produce a conical transmission pattern and polymer transducers for a cylindrical pattern. Our ceramic transducers have a 60-degree beam angle. The polymer transducers have a 360-degree horizontal coverage for the transmitters and 180-degree horizontal coverage for the receivers. With these transducers our bench top prototypes can achieve an effective range of 5 meters[2] with 0.5-centimeter measurement accuracy.

3.3 Establishing and Merging Local Coordinate Systems

The initial locations beacon nodes can be obtained either by manual placement or by automatically establishing a local coordinate system. One method for establishing a coordinate system is to deploy some nodes that are capable of accurate long distance ranging (i.e. long range ultrasound or laser range finders). These nodes establish a local coordinate system as shown in figure 1. In this figure nodes A, B and Γ are equipped with laser range finders. The nodes can communicate with each other and decide on a local coordinate system. One solution is that node A becomes the origin with coordinates $(0, 0)$, while Γ has coordinates $(A\Gamma, 0)$ and B has coordinates $(AB\sin\alpha, AB\cos\alpha)$. Such local coordinate systems can be established at different places inside the network. Later on, nodes coordinate and merge their coordinate systems using coordinate system transformations that place the nodes in a unified coordinate system. This operation was previously done in the field of mobile robotics and it is explained in detail in [3]. Our discussion begins on how to estimate node locations within such coordinate systems.

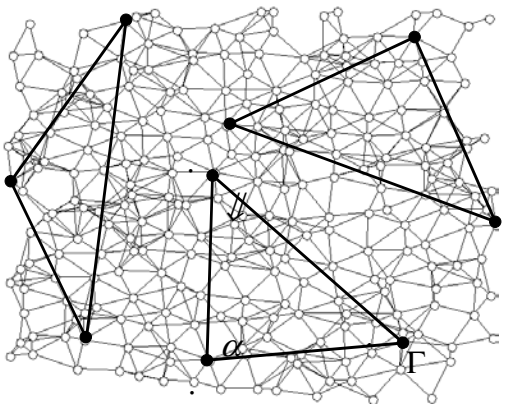


Figure 1: Establishing Local Coordinate Systems

[2]Ranges up to 20m are also achievable but they are not desirable since it would increase multipath effects and increase power consumption

3.4 Sensor Network Viewpoint on Localization

In a sensor network wireless microsensors collaborate with each other to achieve a common sensing task. Such a setup implies that each sensor node views its neighboring sensor nodes as trusted parties, therefore sharing location information does not violate privacy. Furthermore, we view sensor networks as large colonies of sensors that are locally proactive and globally reactive. From this perspective sensor nodes perform local coordination within their neighborhood and then use the locations to perform network scale functions. We consider localization as one of the local coordination functions of the network that takes place during the initial stages of network formation. Later on, during the network lifetime, the acquired knowledge of location is applied to perform tasks such as reporting the origins of events, target tracking, geo-routing and geo-casting, network coverage and network maintenance.

4. SOLUTION OUTLINE

In it simplest form the node localization problem is similar to GPS [12]. If three or more beacons surround an unknown node, the location of the unknown is estimated by minimizing the residuals between the measured and estimated distances between each beacon and the unknown node. In an ad-hoc network since one cannot guarantee that each unknown node will have at least three beacons as neighbors, a different method for estimating node positions needs to be applied. In this paper we extend this basic, single hop multilateration operation performed by GPS to a multihop operation. This operation enables nodes that are not directly connected to a node with known position to collaborate with other intermediate nodes with unknown locations situated between itself and the beacons so that they can jointly estimate their locations. One of the main challenges in this problem is to prevent error accumulation in the network. To prevent this we use least squares estimation to estimate all the unknown node positions simultaneously. In our solution the unknown nodes collaborate to set up a non-linear optimization problem that can then be solved either at a central node or in a fully distributed manner.

The n-hop multilateration takes place in three main phases depicted in figure 2. During the first phase, the nodes form a well constrained or over constrained configuration of unknowns and beacons, the computation subtrees (section 5.1). This configuration forms a system of at least n non-linear equations and n unknown variables to be determined. Computation subtrees also ensure that each unknown member of the computation subtree has a unique possible solution. The nodes that do not meet the criteria for computation subtrees cannot participate in this configuration. The position estimates for such nodes are determined later on in the process. In the second phase, each unknown node computes an initial estimate of its location based on the known beacon locations and the inter-node distance measurements. This is described in section 5.2. The initial estimates obtained in this phase are used to initialize the Kalman Filters of the third phase. The third phase utilizes a Kalman Filter to refine the initial position estimates and compute the final estimate of the node positions (section 6). Finally, the fourth phase uses the computed node estimates to refine the position estimates of nodes that could not participate in the computation tree configuration of the first phase.

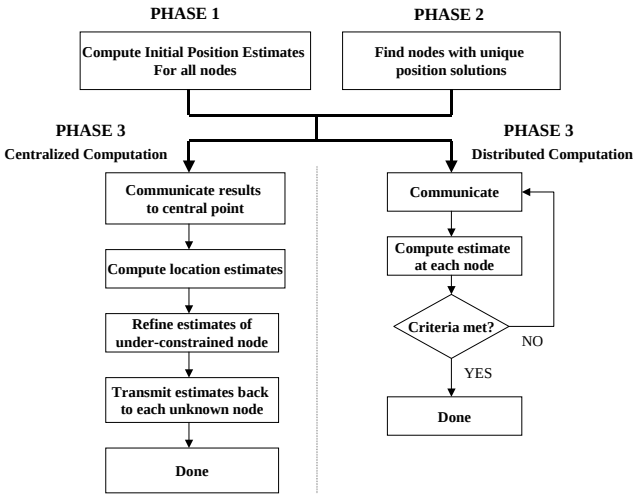


Figure 2: Location Estimation Phases

The first and second phases are independent from each other in an actual network they can take place in parallel. Phases three and four can start as soon as the first two phases are completed and they can terminate at different stages depending on the demands of the application. If computation is done at a central point (either a central computer for the whole network, or a local cluster head), the process will terminate when the unknown nodes receive their position estimates. If the distributed computation form is used, then the processes termination depends on the demands of the application. If the application requires just an indication of proximity, the localization process can only perform the second phase and terminate. If more accurate localization is required the distributed computation will continue until required precision is achieved.

5. INITIAL CONFIGURATION

Before attempting to solve an optimization problem using a Kalman Filter (a gradient descent method for least-squares), one must try to prevent two possible problems may arise: 1) the input problem may have multiple possible solutions and 2) the optimization process may converge at a local minimum. To compute position estimates successfully, one must ensure that the input configuration to the Kalman Filter has a unique solution, otherwise the Kalman Filter may compute a false positive, that is, it will compute an estimate that is numerically correct but it is the wrong estimate of a node's position. In addition to this, if the Kalman Filter is not properly initialized, it may converge to local minima thus a good set of initial position estimates is essential. In this section we describe how our approach avoids these two problems. In order to ensure that the solution is unique, we introduce the creation of computation trees. To avoid converging at local minima we obtain a set of initial estimates that is close to the final estimates using a simple geometrical relationship.

5.1 Phase 1: Computation Subtrees

A computation subtree constitutes a configuration of unknowns and beacons for which the solution to the position estimates of the unknowns can be uniquely determined. This is usually achieved by obtaining a well determined or preferably over-determined set of equations - n variables to be estimated and at least n equations. Before attempting to solve these equations, we first determine a set of requirements that ensure the solution we are about to compute is unique. Computation subtrees have another desirable property that will become apparent when we discuss our distributed computation model in section 6.2. To determine the requirements for solution uniqueness we develop our discussion by reviewing the requirements of the single hop multilateration. Later on, we augment these requirements to cover the multihop case.

5.1.1 One-Hop Multilateration Requirements

In the single hop setup of figure 3a, the basic requirement for one unknown node to have a unique solution on a 2D plane is that it is within range of at least three beacons. If the beacons lie in a straight line, the node configuration is symmetric, and there is more than one possible solution.

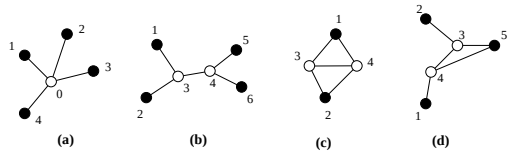


Figure 3: a) One-hop multilateration, b)Two-hop multilateration, c) Symmetric case, d) Each unknown has one independent reference

5.1.2 Two-Hop Multilateration Requirements

Using the one-hop multilateration requirements as a starting point, we establish the corresponding set of requirements for a two-hop multilateration. A two-hop multilateration represents the case where the beacons are not always directly connected to the node but they are within a two-hop radius from the unknown node. In this situation, two or more unknown nodes can utilize the beacon location information and the intermediate distance measurements between themselves and the beacons to jointly estimate their locations. Like the one-hop case, each unknown node needs to be connected to at least three nodes, but these nodes are not required to be beacons. Instead, unknown nodes need to determine which of their neighbors have only one possible position solution and use them as reference points to determine if their position solution is unique. From this perspective, we assume that a node solution is tentatively unique if it has at least three neighbors that are either beacons or their solutions are tentatively unique. Figure 3b illustrates the most basic case. Nodes 3 and 4 are unknown and they are both connected to three nodes. Note that from the perspective of node 3, one of its links terminates to an unknown, node 4. Node 4 however, has two more outgoing links to beacons 5 and 6. If we assume that node 3 has a unique position solution, then node 4 also has a unique position solution. If however, node 4 has a unique position solution, then node 3 is also collaborative because it is connected to 3 collaborative nodes - 1,2 and 4. This condition is necessary but not sufficient to guarantee that there is only one possible node position estimate. Many symmetric topologies that meet the above requirement can yield more than one possible position estimate.

CONDITION 1. *To have a unique possible position solution, it is necessary that an unknown node be connected to at least three nodes that have unique possible positions.*

The fist symmetric case follows from the conditions of the single hop setup - the nodes with tentatively unique solutions used as references for an unknown should not lie in a straight line. If they lie in a straight line, then the unknown node will have two possible positions so the solution to the location estimate is not unique.

CONDITION 2. *It is necessary for an unknown node to use at least one reference point that is not collinear with the rest of its reference points.*

Although the positions of the reference points are not known, one can test for this condition using basic trigonometry. In figure 4, assuming that nodes A,C and D are known to have unique solutions, node B tries to establish if its position solution is unique. To do so node B computes the angles ABC, CBD and ABD. Using the angle ABD, node B can calculate the distance —AD—. If the computed distance AD is equal to the sum of distances AC and CD then the nodes are collinear [3] hence node B decides that its solution is not unique.

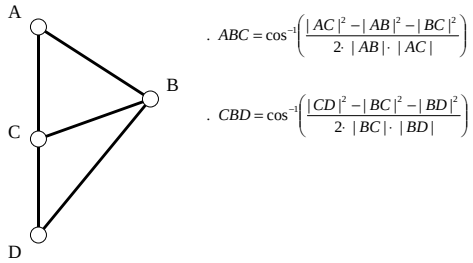


Figure 4: Detecting collinear configurations

Another type of setup that can cause symmetry problems is shown in figure 3c. Nodes 3 and 4 both have 3 links to nodes with tentatively unique positions but the setup is symmetric since the two nodes can be swapped without any violation of the constraints imposed by the intra-node distance measurements. To avoid this situation where the whole network can be rotated over two pivot points (nodes 1 and 2 in this example) we set an additional constraint.

CONDITION 3. *In each pair of unknown nodes that use the link to each other as a constraint, it is necessary that*

[3] Here we loosely use the term 'equal' for clarity and simplicity of the explanation, in practice we also need to consider the noise incurred by the distance measurement process

each node has at least one link that connects to a different node from the nodes used as references by the other node.

The network in figure 3d is an example configuration that satisfies this property. Both unknown nodes 3 and 4 have at least one independent reference. Node 4 has beacon 1 and node 3 has beacon 2. The above three conditions are individually necessary but jointly sufficient to guarantee that if an unknown node is within two hops from at least three beacons then the unknown has a single possible position solution.

5.1.3 N-Hop Multilateration Requirements

To determine if nodes located within n hops from the beacons have unique solutions we use a similar set of criteria as in the single hop case. Starting from the unknown node we test if it has at least three neighbors with tentatively unique positions. If the node has three neighbors that do not already know if their solution is unique, then a recursive call is executed at each neighbor to determine if its position is unique. To meet the requirements of condition 3 each node used as an independent reference is marked. We refer to these nodes as used nodes. This prevents other from subsequent recursive calls to use that node as an independent reference. At every step, each node checks if the criteria for condition 2 are also met. This recursive algorithm is given in figure 5. The algorithm initially starts at one unknown node and constructs a computation subtree by traversing the network in a depth first manner checking for the multilateration requirements. For illustrative purposes, we provide the static version of the algorithm [4]. At each node the algorithm assumes that its caller has a tentatively unique solution and, tries to find out whether the node currently visited has a tentatively unique solution. The algorithm terminates when a computation subtree is formed. In the distributed version of the algorithm, each unknown node only needs to know whether it is a part of the computation subtree, and which of its neighbors are part of the same subtree. Using this information, the node is ready to respond to a call to enter the third phase, position refinement.

5.2 Phase 2: Computing Initial Estimates

Before a Kalman Filter can be started it needs to be properly initialized. To prevent the filter from converging at a local minimum, we apply a simple geometrical relationship based on the accurate ranging measurements taken by the sensors and the knowledge of beacon locations to compute an initial position estimate for each unknown node. The initial estimates are obtained by applying the distance measurements as constraints on the x and y coordinates of the unknown nodes. Figure 6 shows how the distance measurement from two beacons A and B can be used to obtain the x coordinate bounds for the unknown node C. If the distance between an unknown and the beacon A is a then the x coordinates of node C are bounded by a to the left and to the right of the x coordinate of beacon A, $x_A - a$ and $x_A + a$. Similarly, beacon B which is two hops away from C, bounds the coordinates

[4] In practice we apply a distributed algorithm that has multiple starting points and can create a computation tree in a fully distributed manner using by only considering the 2-hop neighborhood of each node. This algorithm is omitted due to space restrictions

Inputs: node id, node id of the caller and a boolean flag that specifies if the caller node is the node that initiated the computation tree search.
Output: if success full a list of edges that make up the computation subtree, otherwise, it returns NULL

```
list isUnique(u, caller, initiator)
begin
if we are at the initiator
  begin U :=0 ; m := 0 end
else begin U := 1; m := 1 end
for each beacon neighbor b begin
    if b is marked begin
      check if we can unmark
      if b is not the only beacon neighbor
      of the node that marked it begin
        mark b as visited by u;
        decrement U for the node that previously
        marked b; U := U + 1
      end
      else begin
        if m < 2 begin
          m := m + 1; U := U + 1;
        end
      end
    end
    else begin
      edge_list ← edge(b, u); U := U + 1
      mark b as visited by u
    end
end
for each unknown marked neighbor z begin
    if m < 2 begin
      edge_list ← edge(u, z)
      m := m + 1 ; U := U + 1
    end
    if U = 3 return edge_list
end
for each unknown unmarked neighbor x
  if (edge_listB := isUnique(x, u, false)) not NULL
    edge_list ← edge(x, u)
    edge_list ← edge_listB
    U := U + 1
    if U = 3 return edge_list
  end
end
return NULL
end
```

Figure 5: Forming computation subtrees

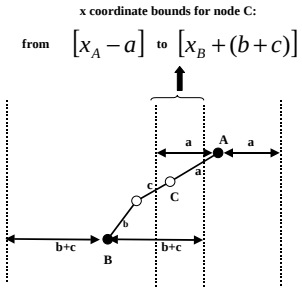


Figure 6: Initial Estimates

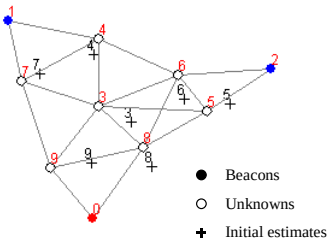


Figure 7: Initial estimates over multiple hops

of C through the length of the minimum weight path to C, $b+c$, so the bounds for C's x-coordinates with respect to B are $x_B-(b+c)$ and $x_B+(b+c)$. By knowing this information C can determine that its x coordinate bounds with respect to beacons A and B are $x_B+(b+c)$ and x_A-a. This operation selects the tightest left hand side bound from and the tightest right hand side bound from each beacon. The same idea is applied on the y coordinates. The node then combines its bounds on the x and y coordinates, to construct a bounding box of the region where the node lies. To obtain this bounding box, the locations of all the beacons are forwarded to all unknowns along a minimum weight path. This forwarding is the same idea as distance vector routing but using the measured distances instead of hops as weights. The initial position estimate of a node is taken to be as the center of the bounding box. When these constraints are combined with the conditions for position uniqueness, they provide a good set of initial estimates for the Kalman Filter used in the next phase. The resulting initial estimates for a 10 node network with 3 beacons is shown in figure 7. One challenge in this method is that the quality of the initial estimates suffers when the beacon-unknown topology is not convex. The unknown nodes that lie within a convex hull created by the beacons can produce good initial estimates. In some configurations, where some of the unknown nodes lie outside the convex hull the initial estimates are still sufficient. In our experiments with large networks we therefore assume that beacons surround the unknown nodes.

6. PHASE 3: POSITION REFINEMENT

In the third phase, the initial node positions are refined, using least-squares estimation. Given that the ranging measurements are noisy this method will give the optimal position estimates in the least-square sense. In our implementation we decided to use a Kalman Filter, which is a recursive solution for least-squares estimation. A Kalman Filter was chosen mostly because of its flexibility and expandability. A Kalman Filter provides a recursive solution to least squares estimation. The main properties of the Kalman Filter that made it an attractive choice are its ability to fuse information from multiple sensing modalities and time prediction based on the previous system state and a system model. These properties are useful for localization, tracking and for other sensing tasks so the Kalman Filter is a valuable component in our infrastructure. The other property of the Kalman Filter that makes it an attractive choice for our application is that it can also track the nodes after the localization process is complete. As described in section 7.4, our experimental node is equipped with sensors that can track the node by dead reckoning once the node's position estimate is known.

The position refinement phase has two possible formulations, centralized and distributed. The first one computes many unknown estimates at a single point (i.e a central node a locally central cluster head). The second is a distributed approximation of the central implementation in which each node iteratively refines its position with local computation and communication.

6.1 Computing at a Central Node

Using the computation subtrees and the initial position estimates, we can compute the initial node estimates at a central point. The edges of the computation subtree give a well-determined or over-determined set of equations, which can be solved using non-linear optimization. The non-linear of equations for the network in figure 3b is shown in equations 1 [5]. As in the one hop case, the objective is to minimize the residual between the measured distances between the nodes and the computed estimates, which are the result of the estimation process.

$$\begin{aligned} f_{2,3} &= R_{2,3} - \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\ f_{3,5} &= R_{3,5} - \sqrt{(ex_3 - x_5)^2 + (ey_3 - y_5)^2} \\ f_{4,3} &= R_{4,3} - \sqrt{(ex_4 - ex_3)^2 + (ey_4 - ey_3)^2} \\ f_{4,5} &= R_{4,5} - \sqrt{(ex_4 - x_5)^2 + (ey_4 + y_5)^2} \\ f_{4,1} &= R_{4,1} - \sqrt{(ex_4 - x_1)^2 + (ey_4 - y_1)^2} \end{aligned} \quad (1)$$

The $R_{i,j}$ quantities represent the measured distances between two nodes and the quantities under the square root indicate the estimated distances. $f_{i,j}$ represent the residual between the measured and estimated quantities. The objective function in 2 is to minimize the mean square error over all equations. The difference of this from its one hop counterpart is that in this process, unknown-unknown links are also used.

$$F(x_3, y_3, x_4, y_4) = min \sum f_{i,j}^2 \quad (2)$$

[5]the prefix e in front of x, y denotes estimated coordinates, as opposed to known coordinates

The solution of these equations using a Kalman Filter is described in the following subsection.

6.1.1 Kalman Filter Implementation

A Kalman Filter consists of two phases, a time update phase and a measurement update phase. The former is a prediction in time (equations 3, and 4). This time prediction $\hat{x}_k^-$ is based on a known model of the system behavior, represented by matrix A. u_k is a zero mean gaussian random variable and B is the error covariance matrix for this random variable. P_k^- is the apriori estimate of the error covariance and Q is the process noise. The latter is an update of the current time estimate based on a measurement that was just obtained. K represents the Kalman Filter gain and it serves a weight to the residual of the filter. The residual is the difference between the measurement, (represented by z_k) and the predicted measurement $H\hat{x}_k^-$. $\hat{z}_k$ the distance between nodes, based on the current position estimate. Matrix H is the Jacobian of $\hat{z}_k$ with respect to the apriori estimates (found in $\hat{x}_k^-$) of the locations. Matrix R is the measurement noise covariance matrix. This contains the known noise covariance of the distance measurement system (i.e based on the characterization of our ultrasonic system we assume white gaussian noise with standard deviation = 20 mm). $\hat{x}_k$ is the new estimate obtained after the prediction and measurement are combined. This new prediction measurement has a new error covariance matrix P. Matrix I stands for the identity matrix. For a good introduction discussion of Kalman Filters we refer the reader to [18] and for a more in-depth discussion we recommend [19].

For the purposes of the n-hop multilateration primitive, we assume that the network is static. Since the positions of the nodes, do not change in time, the time update phase is not used. In fact, the result of the Kalman Filter would be the same as the result of iterative least squares [15]. Based on this we focus our discussion on the second part of the Kalman Filter, the measurement update phase.

$$\hat{x}^- = A\hat{x}_{k-1} + Bu_k \tag{3}$$

$$P_k{}^- = AP_{k-1}A^T + Q \tag{4}$$

$$K = P_k{}^- H^T (HP_k{}^- H^T + R)^{-1} \tag{5}$$

$$\hat{x}_k = K_k(z_k - H\hat{k})^{-1} \tag{6}$$

$$P_k = (I - K_k H)P_k^- \tag{7}$$

To estimate the unknown locations, our algorithm proceeds as follows:

1. Set the vector to the initial estimates obtained in section 5.2
2. Evaluate equations 5, 6 and 7 - the measurement update phase
3. Evaluate the convergence criterion $\sqrt{(\hat{x}_k)^2 - (\hat{x}_k^-)^2} \leq \Delta$ where Δ is some predefined tolerance (0.01 in our experiments). If the criterion is met then the algorithm terminates and has the new position estimates. Otherwise,
4. Set the prediction $\hat{x}_k^-$ to the new estimate $\hat{x}_k$ and restart from step 2.

The term Δ is used as an indicator of the gradient of the Kalman Filter and shows how far the process is from convergence. For illustrative purposes we provide the setup of the Kalman Filter in terms of 3c. The initial estimates (denoted by ex and ey) are placed in vector, $\hat{x}_k^-$, so $\hat{x}_k^- = [ex_3, ey_3, ex_4, ey_4]$. The ranging measurements are placed in vector z_k, $z_k = [R_{2,3}, R_{3,5}, R_{3,4}, R_{4,1}, R_{4,5}]$. Vector $\hat{z}_k$ contains the ranging distances based on the current estimates

$$\hat{z}_k^T = \begin{bmatrix} \sqrt{(x_2 - ex_3)^2 + (y_2 - ey_3)^2} \\ \sqrt{(ex_3 - x_5)^2 + (ey_3 - y_5)^2} \\ \sqrt{(ex_3 - ex_4)^2 + (ey_3 - ey_4)^2} \\ \sqrt{(ex_4 - x_1)^2 + (ey_4 - y_1)^2} \\ \sqrt{(ex_4 - x_5)^2 + (ey_4 - y_5)^2} \end{bmatrix}$$

Matrix H is the jacobian of $\hat{z}_k$ with respect to $\hat{x}_k^-$.

$$H = \begin{bmatrix} 0 & 0 & \frac{x_2 - ex_3}{\hat{z}(1)} & \frac{y_2 - ey_3}{\hat{z}_k(1)} \\ \frac{ex_3 - x_5}{\hat{z}_k(2)} & \frac{ey_3 - ey_4}{\hat{z}_k(2)} & 0 & 0 \\ \frac{ex_3 - ex_4}{\hat{z}_k(3)} & \frac{ey_3 - ey_4}{\hat{z}_k(3)} & \frac{ex_4 - ex_3}{\hat{z}_k(3)} & \frac{ey_4 - ey_3}{(z)_k(3)} \\ \frac{ex_4 - x_1}{\hat{z}_k(4)} & \frac{ey_4 - y_1}{\hat{z}_k(4)} & 0 & 0 \\ \frac{ex_4 - x_5}{\hat{z}_k(5)} & \frac{ey_4 - y_5}{\hat{z}_k(5)} & 0 & 0 \end{bmatrix}$$

The above illustration shows how the matrix size and subsequently the amount of computation increases with the number of nodes. Each edge in the collaborative subtree contributes one entry in the measurement matrix z_k. In matrix H, the number of unknown nodes determines the number of columns and the number of edges determines the number of rows. Each beacon-unknown edge adds two entries to the row and each unknown-unknown edge adds four entries. The noise covariance matrices P_k^- and P_k are square matrices whose size depends on the number of unknown nodes and the measurement noise matrix R is a square matrix and it size determined by the number of edges in the collaborative subtree. These changes in matrix sizes dramatically increase the amount of computation that has to be performed. Furthermore, from our simulation experience, we noted that when the Kalman Filter has more variables to estimate, it takes more iterations to converge. Unfortunately, since the estimation process is an iterative process we cannot quantify the amount of computation required for the filter to converge analytically. Instead, we evaluate this empirically in our simulations by measuring the number of FLOPS MATLAB consumes [6] until the Kalman Filter converges. This

[6] Note that the FLOPS measure obtained from MATLAB

description also shows that although node positions can be estimated accurately using this method, such computation cannot be performed using a low cost microcontroller available on the sensor nodes. To facilitate this type of computation, we have developed a distributed approximation to this method - distributed n-hop multilateration.

6.2 Computing at Every Node

In the distributed version, of our algorithm, computation is spatially distributed across the network and each unknown node is responsible for computing its own location. This is achieved by performing local computation and communication with the neighboring nodes. This idea of using a decentralized Kalman Filter to distributed computation across nodes is not new. Durrant Whyte presented a decentralized Kalman Filter in [22]. Roumeliotis [13] is also applying a distributed Kalman Filter in his synergetic localization scheme. The novelties of our scheme that makes it different from the previous approaches are:

- The Kalman Filter executes in the context of a computation subtree. As we will describe in this section, this enables every unknown node to obtain its global optimal position estimate locally.
- We use an approximation of the Kalman Filter instead of the full form in order to conserve computation and we provide a scalability comparison of centralized vs. distributed in terms of computation overhead. This is crucial for the viability of our approach since we want to run this on resource-constrained devices.
- The computation is applied over a multihop network thus it is inherently related the concepts of ad-hoc networks, in fact, the estimation itself is a least-squares-communication hybrid.

The underlying principle of our distributed scheme is that after the completion of phases one and two, each node inside the computation tree computes an estimate of its location. Since most unknown nodes, are not directly connected to beacons, they use the initial estimates (obtained in section 5.2) of their neighbors as the reference points for estimating their locations. As soon as an unknown computes a new estimate, it broadcasts this estimate to its neighbors, and the neighbors use it to update their own position estimates. This computation is repeated from node to node across the network until all the nodes reach the pre-specified tolerance Δ, $\left(\sqrt{(\hat{x}_k)^2 - \left(\hat{x}_k^-\right)^2} \leq \Delta\right)$. Figure 8 is a pictorial representation of the computation process. First node 4 computes its location estimate using beacons 1 and 5 and node 3 as references. Once node 4 broadcasts its update, node 3 recomputes its own estimate using beacons 2 and 5 and the new estimate received from node 4. Node 3 then broadcasts the new estimate and node 4 uses this to compute a new estimate that is more accurate than its previous estimate.

is used for relative comparison of the computation cost between our centralized and distributed schemes. In our actual implementation, the Kalman Filters are implemented in C and they execute on the node processor.

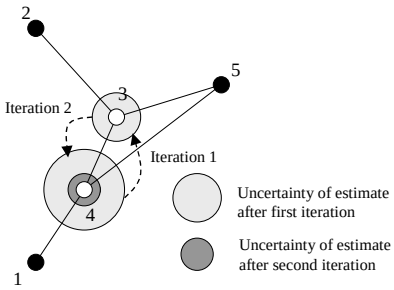


Figure 8: Initial estimates over multiple hops

If this process proceeds uncontrolled, then the nodes will converge at local minimal and erroneous estimates will be produced. Imagine a computation subtree with many unknown nodes (i.e 20). If two neighboring unknown nodes A and B that compute and broadcast their updates as soon as an update from each other is received, then their updating process will proceed faster than the remaining nodes in the computation subtree. This introduces a "local oscillation" in the computation that makes the nodes converge to their final estimates much faster but without complying with the global gradient. Because of this the nodes will produce incorrect position estimates.

To prevent this problem, the Kalman Filters at each node are executed in a sequence across all the unknown members of the computation subtree. This sequence is repeated until the Kalman Filters of all the members of the computation subtree converge to a pre-specified tolerance. The in-sequence execution of the Kalman Filters inside the computation subtree establishes a gradient with respect to the global topology constraints at each node, thus enabling the node to compute its global optimum locally.

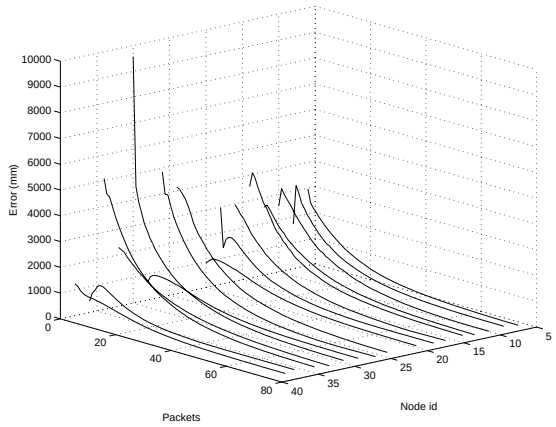


Figure 9: Progress of distributed computation

Figure 9 is an excerpt from our combined ns-2-matlab simulation. As we will describe in the next section, we developed our simulations in ns-2. To be able to measure FLOPS, we implemented the Kalman Filters in MATLAB and then using the MATLAB compiler and the MATLAB C++ libraries, we translated the filters to C++ so we can use them

at the routing layer in ns-2. The figure demonstrates the execution of distributed position refinement on a network of 34 nodes and 6 beacons [7]. The unknown nodes in the network have an average degree of 4, 15 meters range and the 20 mm white gaussian noise in the distance measurement system. The x-axis shows the number of packets (estimate updates) transmitted by each node, the node ids are shown in the y-axis and the z-axis shows the error in millimeters. The values of the error before any packets are transmitted, at packets = 0 on the x-axis, represent the errors of the initial estimates (obtained in section 5.2). As it can be seen form the figure, each node starts at different levels of error. After a few iterations of executing the node sequences in the computation subtree, a global gradient is established that drives the error down across the whole subtree. In the end, each node succeeds in estimating its node location with a 3-centimeter accuracy. The fact that error accumulation is eliminated is due the properties of the computation subtree. As explained in 5.1, computation subtrees constitute a constrained configuration of nodes and beacons. This constrained configuration prevents the estimates from going in the wrong direction.

The order of nodes executing in the computation subtree sequence does not need to be specified but it needs to be consistent over successive iterations of the sequence. This entails that the order with which nodes compute their position updates has to be consistent across iterations. One possible way to initiate this distributed computation process is by using Distributed Depth First Search (DDFS). DDFS search is started at an arbitrary unknown node within the computation tree and its run for two iterations. During the traversal of the subtree by DDFS, when each node is marked visited, the node it computes and broadcasts its location estimate and starts a timer. In the second iteration of DDFS, nodes compute and broadcast their location. At this point the nodes also stop the previously set timer. The time between the two visits denotes the time interval at which each node should recompute and broadcast a new position update. The distributed algorithm for driving the distributed computation process is shown in figure 10. Figure 9 shows how the computation proceeds on a network of 6 beacons and 34 unknowns. The DDFS algorithm for this example was obtained from [21].

Finally, note that the Kalman Filter used in our experiments is an approximation of the distributed Kalman Filter. The results of the two methods can be made equivalent if nodes also exchange and update their corresponding covariance matrices, as it is done in [13]. In our implementation, we do not communicate the covariance matrices to the neighboring nodes, to conserve computation and communication. Before finalizing this design decision we verified that the results obtained by this approximation do not compromise the overall quality of our estimates. We tested this by comparing the outputs of our centralized and distributed position refinement phases. The tests where run on a test suite of 42 networks of different sizes varying from 10 to 100 nodes. From this comparison we found out that the difference in the results between the centralized Kalman Filter the distributed approximation we used is very small. Figure 11 depicts the result of the comparison. The mean difference is 0.015 millimeters with a standard deviation of 0.54mm. Based on this result we verified that our distributed approximation of the Kalman Filter does not compromise our computation accuracy.

[7]For clarity and good visibility purposes the graph only shows how the process proceeds on the even numbered nodes.

Start the algorithm (initiator only!)
$visited_u := true$
for i:=1 : 2
  for $w \in Neigh_u$
    do begin send ⟨**bfs**⟩ to w; $status_u[w] := cal$ **end**

Upon receipt of ⟨**bfs**⟩ from v:
if not $vistited_u(i)$ **then**
  begin
    $visited_u(i)$:=$true$;$status_u(i)[v]$:=$father$
    compute new location estimate and broadcast it
    if i=0 $t1$:=time(); i:=$i+1$
    else $updatePeriod$ = time()-$t1$;
    timer.schedule($updadePeriod$)
  end
if $status_u(i)[v]$=$unused$ **then**
  begin send ⟨**bfs**⟩ to v;$status_u(i)[w]$:= ret **end**
else if there is a w with $status_u(i)[w]$:=$unused$ **then**
  begin send ⟨**bfs**⟩ to w $status_u(i)[w]$:=cal **end**
else if there is a w with $status_u(i)[w]$=$father$ **then**
  begin send ⟨**bfs**⟩ to w **end**
else (* initiator *) **stop**

Upon a timeout:
if $converged = false$ **or** $update\_needed = true$
  begin
    compute a new location estimate and broadcast it
  end
if $\sqrt{(\hat{x}_k)^2 - (\hat{x}_k^-)^2} > \Delta$
  begin $converged := true$ **end**
begin timer.schedule($updadePeriod$) **end**

Upon receipt of a updated location broadcast:
  if $converged = true$
    begin $update\_needed := true$ **end**

Figure 10: Distributed computation algorithm driven by DDFS

7. EVALUATION

We evaluate the performance of the n-hop multilateration primitive through a set of simulations. The Kalman Filters are implemented in MATALB and they are linked into the ns-2 simulator using the MATLAB compiler and the MATLAB C++ library. The required protocols for communication are implemented inside ns-2. Using this simulation setup we carried out a series of experiments on a test suite of 200 different scenarios. Our simulation parameters are set to match the parameters of our experimental node. Each node has an effective radio range of 15 meters and each node can measure distances between its neighbors with the same range as the radio. The measurement noise is modeled as a

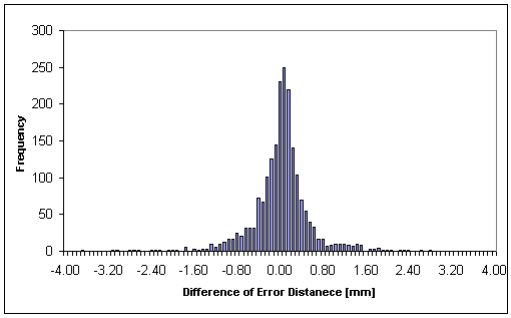


Figure 11: Comparison of centralized and distributed outputs

zero mean gaussian random variable with 20 mm standard deviation.

The primary goal of our ns-2 implementation is to verify the correct operation of our distributed computation scheme over a wireless ad-hoc network. The distributed algorithm version of the n-hop multilateration is implemented as a routing layer in ns-2. This routing layer also contains a minimal protocol that discovers the two-hop neighborhood of each node, and a forwarding mechanism for forwarding packets with beacons locations as described in section 5.2. These minimal routing requirements are sufficient to form computation subtrees in a fully distributed fashion, to obtain the initial estimates and to perform the distributed localization process. At the MAC layer we use a modified version of the IEEE 802.11 protocol with a 15-meter transmission range and an effective data rate of 20kbps.

For the centralized case, we used DSR as the routing protocol, IEEE 802.11 as the MAC and we run the Kalman Filter at the application layer.

7.1 Computation Cost Comparison

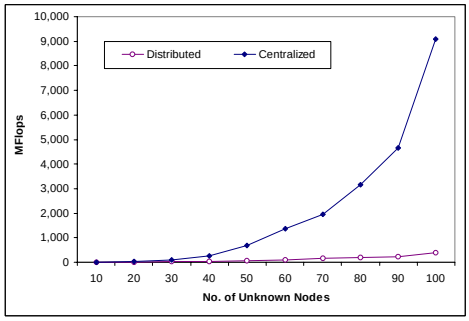


Figure 12: Computation cost comparison

Our first experiment compares the computation overhead between the distributed and centralized computation methods by recording the number of FLOPS consumed by MATLAB to compute the position estimates in each case. The scenarios used for this test have 6 beacons and varying number of unknowns ranging from 10 to 100 nodes. The number of unknowns was used in increments of 10, and the results show the average for 20 scenarios of each type. In all cases the network density is kept constant and each node has an average of 6 neighbors. The cumulative number of MFLOPS for the centralized and distributed implementation are shown in figure 12. From this result, we found that the computation overhead of the centralized computation model increases fast with the number of unknown nodes. In this particular test, the computation overhead appears to be cubic with the number of nodes. The distributed computation model on the other hand scales linearly with the number of nodes. The slope for the distributed case in figure 12 is 3.7MFLOPS, meaning that each node spends approximately 3.7MFLOPS to compute an estimate of its location. Network density has a similar trend on computation overhead. As the network density increases, the amount of computation required by the centralized model increases exponentially. Figure 13 shows an example of this case. In this example, the number of unknowns and beacons in the network is kept constant. Changing the area of the network varies the network density. The amount of computation at each node in the centralized case is much higher with respect to the number of unknown nodes n. In the distributed case, density only increases the number of edges at each node hence the computation scales linearly with the number of nodes.

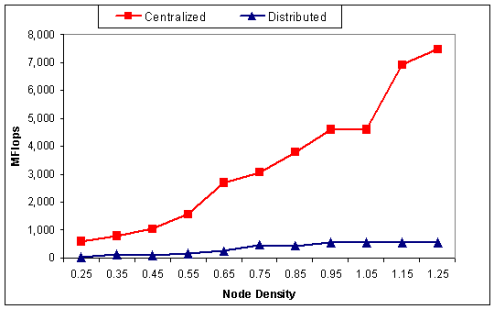


Figure 13: Computation vs. network density

From this comparison we conclude that the distributed computation model is a better choice for computing node locations. Even if computation is performed a central point, using the distributed model to compute locations will help to reduce computation latency. This is especially important in the case of low cost processors that do not have hardware support for floating point operations. The AT91FR4081 microcontroller of our localization node is one such example. The ARM7TDMI core of our microcontroller has a fixed-point multiplier, so we perform the floating-point operations of our Kalman Filter using a software floating-point library. As a second level of optimization, we are now in the process of implementing the Kalman Filter using fixed-point arithmetic on our experimental node.

7.2 Localization Accuracy

To quantify the accuracy of the localization error we applied two tests. The first test evaluates accuracy of the localization process based on the measurement noise parameters of our ultrasonic distance measurement system. Figure 14 shows how the error in the estimates increases as the network scales. The ratio of beacons with respect to the unknowns

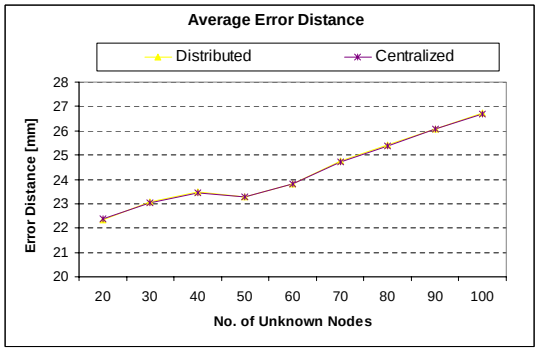


Figure 14: Quality of localizations a unknown nodes increase

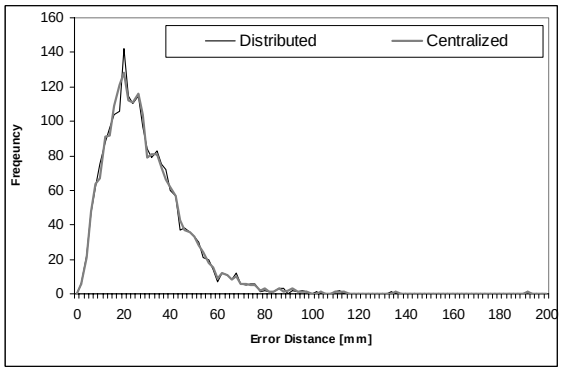


Figure 15: Estimate error distribution

is kept constant at 20 percent. The error in the estimates increases very gracefully as the network scales.

Figure 15 shows the cumulative error distribution over all scenarios used in this experiment for both the distributed and centralized cases. In both cases the average error was 27.7 millimeters with a standard deviation of 16 mm.

We observed a similar trend when we computed the error in the estimates at different measurement noise levels. This reflects how the n-hop multilateration would perform with less accurate distance measurement systems as the percentage of beacons descreases. The results of our simulation are shown in figure 16, and a based on different network sizes ranging from 10 unknown nodes to 100 unknown nodes, 20 networks for each set. In all cases the number of beacons is kept constant, 6 beacons were used in each network. In this particular test we also found that the performance of the distributed version degrades faster in networks of more than 100 nodes. This occurred in cases where the Kalman Filter sequence started by estimating the locations of nodes with accurate initial estimates using neighboring unknown nodes with less accurate initial estimates. One possible approach to mitigate this effect is to take the size of the bounding box (computed in section 5.2)into account. We expect that if the Kalman Filter computation sequence starts at the nodes with the largest bounding boxes then the problem mentioned above could be avoided and the convergence will be faster. We plan to explore this as part of our

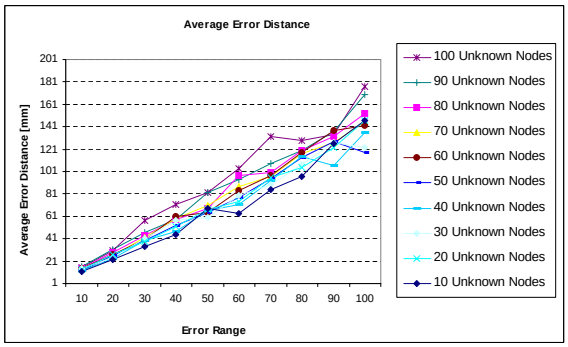


Figure 16: Errors in estimates at different measurement noise levels

7.3 Communication Cost and Convergence Latency

The convergence latency and communication aspects of the n-hop multilateration are more difficult to evaluate because of their dependence on multiple system attributes. In the fully distributed case, convergence latency depends on the available communication bandwidth and the node processing power. Convergence latency also depends on the size of the computation tree. As the number of nodes increases, the sequence of Kalman Filter executions will take longer to complete and more iterations of the sequence are required. The communication pattern is uniform across all the nodes.

When computing at a single point in the network, the convergence latency of the n-hop multilateration primitive is a function of the communication latency for transmitting the packets from each member of the computation subtree and back, and also depends on the power of the central processor. If the computation tree is large, the computation latency will be the dominant component. As an example a network of 100 unknowns and 6 beacons takes approximately 5 to 7 minutes to converge on our Pentium III 700 MHz workstation. Evaluating the communication cost is more complex. In a clustered architecture, the communication cost depends on the cost of electing a cluster head and the cost to propagate the information back and forth from the cluster head.

Figure 17 is a comparison of the communication cost of the n-hop multilateration process in the centralized and distributed case, executed on a network of 44 unknown nodes and 6 beacons. The figure shows the total number of bytes transmitted by each node during the n-hop multilateration process. The average number of bytes transmitted is 4596 for the centralized scheme and 4485 for the distributed scheme. Although on average the communication cost is almost the same, the distributed scheme has an even distribution of transmitted bytes. Additionally, a favorable trade of the distributed version on the n-hop multilateration is shown in figure 18. The average convergence latency for two networks sizes of 20 and 40 nodes is shown. During the latest part

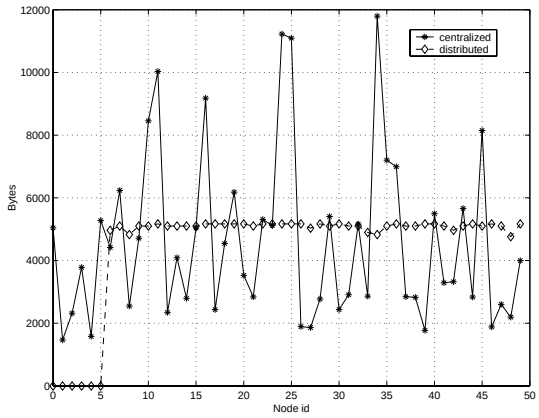


Figure 17: Communication cost on a 50 node network (6 beacons, 44 unknowns)

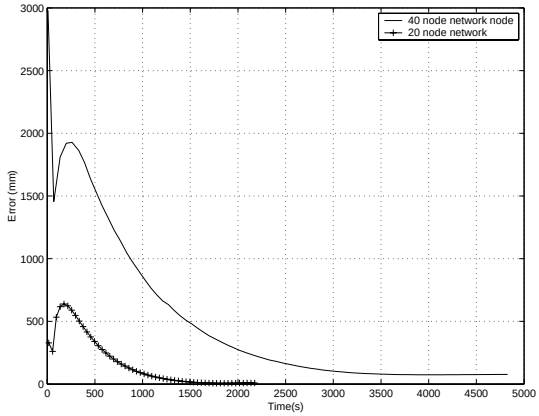


Figure 18: Convergence latency of distributed n-hop multilateration

of the process, a lot of computation is spent to achieve a relatively small refinement of the position estimate. At this point, higher-level applications should have the option to decide when to stop the position refinement phase by adjusting the convergence criterion Δ. Finally, we note that this process is robust since the position refinement phase can continue even if some of the nodes fail during the process.

7.4 Localization Experimental Node

To experiment with different types of node localization problems we developed a wireless sensing device that is geared towards experimentation. The wireless communication front end of this device is similar to UC Berkeley's Rene and MICA motes **??** and UCLA's Medusa node described in [2]. It features a low power radio from RF Monolithics and a 4MHz ATMega103L microcontroller from ATMEL. The design decisions for the remaining part of the node are however different from the nodes mentioned above. In addition to the ATMega103L microcontroller, the node has a 40MHz ARM THUMB microprocessor, also from Atmel. This serves as

Figure 19: Our experimental node

a more powerful computation co-processor[8]. This processor has 1MB of FLASH memory and 136KB of RAM, which is sufficient to run several existing embedded operating systems such as Red Hat's eCos and uCLinux. This provides a versatile programming environment for studying the interaction of our localization algorithms with existing protocols currently used by a wide variety of wireless devices.

In addition to the more powerful processor, our research node has 2 pushbuttons that can be used as user interfaces, and an RS-485 transceiver. This provides a 2MBps bus to facilitate data collection in experiments or to server as a gateway to the infrastructure. The RS-485 link also enables the formation of node arrays (up to 36 nodes can be daisy-chained together) help with experimentation and data collection in various settings. Our node supports several power saving modes and contains three current monitors that can monitor the power consumption of the different component of the node. The main board also carries a MEMs accelerometer that detects if the node has moved during the localization process. Later on we will also use the accelerometer as another sensing modality in our Kalman Filter to track the nodes once the localization process is completed. The distance measurement front end comes on an accessory board that carries an array of 40KHz ultrasonic sensors, and a magnetometer that can serve as a compass. The ultrasonic distance measurement system has an effective range of 5 meters and an accuracy of 0.5 centimeters. The node is power by a single 550mAh Lithium-Ion rechargeable battery. The circuit boards were customized to fit in a 2 x 3 x 1.25 epoxy enclosure shown in the figure.

8. CONCLUSIONS AND FUTURE WORK

In this paper we have described, the n-hop multilateration primitive for node localization problems. We have shown that using this three phase approach nodes that are indirectly connected to nodes with node locations can estimate their locations with similar accuracies at the single hop multilateration. Also, with our distributed approach colonies of constrained sensor nodes can collectively solve a global op-

[8]At design time we considered that Moore's law will prevail so we decided to go with a bigger processor that has more memory and a multiplier. This proved to be a good design choice since the new replacement part for the ATMega103L, ATMega128L does have a multiplier

timization problem that an individual node cannot solve. The use of a global gradient for computing a global optimum locally reinforces a distributed computation model with other potential applications in sensor networks. Our simulations have shown that although the computation cost scales linearly with the number of nodes, the increasing amount of communication suggests that the n-hop multilateration primitive should be applied hierarchically in large networks. As we have explained at the beginning of this paper, a good starting point for such a hierarchical is to establish sevaral coordinate systems locally using long-range distance measurements and then perform the required coordinate transformations to map the entire network to a single coordinate system. In addition to the distributed computation model the n-hop multilateration primitive appears to be an attractive choice for assisting infrastructure based localization systems to better handle obstructions. In this paper we have developed the computational part of the n-hop multilateration. The remaining challenge is to study its feasibility with respect to the physical effects. To this end, as part of our future work we plan to study the interaction of our algorithms with the physical world through the localization node that we have developed.

9. REFERENCES

[1] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, *Next Century Challenges: Scalable Coordination in Sensor Networks*, Proceedings of the fifth annual international conference on Mobile computing and networking, Seattle, Washington, 1999, Pages: 263 - 270

[2] A. Savvides, C. C. Han and M. B. Srivastava *Dynamic Fine-grained Localization in Ad-Hoc Networks of Sensors,* Proceedings of the fifth annual international conference on Mobile computing and networking, Mobicom 2001, Rome, Italy, July 2001, pages 166-179

[3] A. Howard, M. J Mataric and G. S. Sukhatme, *Relaxation on a mesh: a formalism for generalized localization,* Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS01), pages 1055–1060, 2001

[4] Intersense Inc http://www.isense.com

[5] Lewis Girod and Deborah Estrin , *Robust range estimation using acoustic and multimodal sensing* Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), Maui, Hawaii, October 2001.

[6] N. Priyantha, A Chakraborthy, and H. Balakrishnan, *The Cricket Location Support System, Proceedings of International* Conference on Mobile Computing and Networking, pp. 32-43, August 6-11, 2000 Boston, MA

[7] J. Li, J Jannotti, D.S J. DeCouto, D. R. Karger and R. Morris, *A Scalable Location Service for Geographic Ad-Hoc Routing*, Proceedings of International Conference on Mobile Communications and Networking, August 11-16, Boston, Massachusetts

[8] Y. Xu, J. Heidemann and D. Estrin, *Geography-informed Energy Conservation for Ad Hoc Routing* , Proceedings of the ACM SIGMOBILE 7th Annual International Conference on Mobile Computing and Networking, Rome, Italy, July 2001

[9] S. Meguerdichian, F. Koushanfar, G. Qu, M. Potkonjak, *Exposure In Wireless Ad Hoc Sensor Networks*, International Conference on Mobile Computing and Networking (MobiCom '01),pp. 139-150, Rome, Italy, July 2001..

[10] L. Doherty, L. El Ghaoui, K. S. J. Pister, *Convex Position Estimation in Wireless Sensor Networks*, Proceedings of Infocom 2001, Anchorage, AK, April 2001.

[11] SICK http://www.sick.de

[12] E. Kaplan, *Understanding GPS Principles and Applications* Artech House, 1996

[13] Roumeliotis, S.I.; Bekey, G.A. *Synergetic localization for groups of mobile robots*, Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, NSW, Australia, 12-15 Dec. 2000.) Piscataway, NJ, USA: IEEE, 2000. p.3477-82 vol.4. 5 vol. (lxiii+li+5229)

[14] R. Wand, A. Hopper, V. Falcao and J. Gibbons, *The Active Bat Location System*, ACM Transactions on Information Systems, January 10 1992, pages 91-102

[15] E. Foxlin, M. Harrington, and G. Pfeiffer *Constellation(tm): A Wide-Range Wireless Motion-Tracking System for Augmented Reality and Virtual Set Applications*, Proceedings of Siggraph 98, Orlando, FL, July 19 - 24, 1998

[16] P. Bahl, V. Padmanabhan, RADAR: *An In-Building RF-based User Location and Tracking System*, Proceeding of INFOCOM 2000 Tel Aviv, Israel, March 2000, p775-84, vol 2

[17] N. Bulusu, J. Heidemann and D. Estrin, *GPS-less Low Cost Outdoor Localization For Very Small Devices*, IEEE Personal Communications Magazine, Special Issue on Networking the Physical World, August 2000

[18] G. Welch and G. Bishop *An Introduction to the Kalman Filter* Available from http://www.cs.unc.edu/ welch/kalman/kalmanIntro.html

[19] R. Brown and P. Hwang *Introduction to Signals and Applied Kalman Filtering* Wiley Press 1997

[20] Hexamite http://www.hexamite.com

[21] G. Tel *Distributed Graph Exploration* Obtained from http://carol.wins.uva.nl/ de-laat/netwerken_college/explo.pdf

[22] BS Rao and HF Durrant-Whyte, *Fully Decentralized algorithm for multisensor Kalman filtering* IEE Proceedings-D, Vol. 138, No.5 September 1991

[23] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister *System Architecture Directions for Networked Sensors"*, Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS -IX), pages 93-104, Cmbridge, MA, USA, Novemeber 2000

Appendix O:

A Distributed Computation Platform for Wireless Embedded Sensing

Andreas Savvides and Mani B. Srivastava
Networked and Embedded Systems Lab
Electrical Engineering Department
University of California, Los Angeles
{asavvide, mbs}@ee.ucla.edu

Abstract

We present a low cost wireless microsensor node architecture for distributed computation and sensing in massively distributed embedded systems. Our design focuses on the development of a versatile, low power device to facilitate experimentation and initial deployment of wireless microsensor nodes in deeply embedded systems. This paper provides the details of our architecture and introduces fine-grained node localization as an example application of distributed computation and wireless embedded sensing.

1. Introduction

The rapid advancements in embedded wireless devices have enabled a new set of interesting and diverse applications. One class of applications is wireless microsensor networks where small devices embedded in the environment coordinate with each other to perform unsupervised sensing and actuation. Typical tasks include condition-based maintenance in factories, monitoring remote ecosystems, endangered species, forest fires and disaster sites [9]. To complete their sensing tasks, in tiny wirelessly connected sensor nodes are required to form an ad-hoc network that can sense events, interpret the sensor readings and report the results to a remote control center. This paradigm creates a set of new multidisciplinary challenges that need to be addressed. First, new lightweight and energy efficient methods are required to enable nodes to self-organize and construct a network on the fly are required. Second, the different sensing modalities need to be well understood. Third, new mechanisms for the in-network processing of sensor data need to be developed to improve system latencies and help to conserve power by reducing communication.

In our efforts to develop robust wireless sensor networks that can operate without human supervision, we study the operation of sensor nodes under realistic deployment conditions by constructing a deeply embedded wireless microsensor system. As a vehicle to the exploration of such systems, we have developed the Medusa MK-2 node (Figure 1), a versatile, low cost, low power wireless sensing device. The goal of this device is to enable experimentation with different sensing technologies, assist with the development of new sensor network protocols and applications and to accommodate the first deployment phase of a deeply embedded sensing environment, the Smart Kindergarten [3]. In the context of our research, the Medusa MK-2 node is used to provide fine-grained node localization services in the Smart Kindergarten environment for studying group interaction problems, but can also provide a flexible platform for the study of a wide variety of applications.

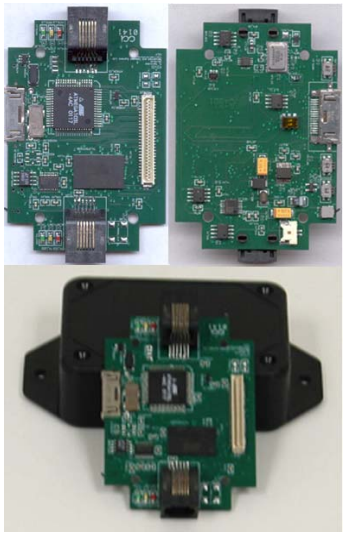


Figure 1 The Medusa MK-2 node

This paper presents the details of the Medusa MK-2 design. While our design focuses on producing a low cost low power distributed computation platform for wireless embedded sensing, great care is taken to attain the maximum flexibility for experimentation. The remainder of this paper is organized as follows. The next section provides an overview of some general sensor node requirements and introduces the MK-2 architecture. Section 3 provides a detailed description of the node subsystems. Section 4 discusses node localization as an example application, section 5 presents the related work and section 6 concludes the paper.

2. Sensor Node Requirements

In typical sensor network scenarios, large numbers sensor nodes are expected to be deployed an ad-hoc manner to monitor a set of events [9]. The design of such sensor nodes is driven by the following factors:

- **Size –** In order to be unobtrusive to their environment these nodes should have a small form factor.
- **Cost –** These devices are expected to be deployed in large number and be disposable. This implies that they should be manufactured with very low cost.
- **Power Efficiency –** Since these devices are expected to be small, they should be able to operate over small batteries for prolonged time periods. To do so sensor nodes should use low power components and also try to make optimal use of the available energy resources.
- **Flexibility –** To facilitate experimentation these devices should be very flexible in programmability and should have a rich set of hardware and software sensor interfaces to accommodate different sensing technologies.

Figure 2 depicts a typical sensor node architecture. The node consists of a power supply subsystem that contains a battery and a DC-DC converter, a processing unit which is usually made up of a low cost, low power microcontroller, some memory, a set of sensors and a low power radio for communicating with other nodes. To optimize the operation of a sensor network made of such nodes all components attributing to the operation of the sensor node need to be closely studied and understood.

2.1 Medusa MK-2 Overview

To facilitate our research and experimentation in sensor networks we have developed the Medusa MK-2 wireless sensor node. Although the primary driver for the development of this node is the study of node localization problems, Medusa MK-2 is also a versatile device for testing different sensing solutions and for exploring a wide variety of new protocols and applications in sensor networks.

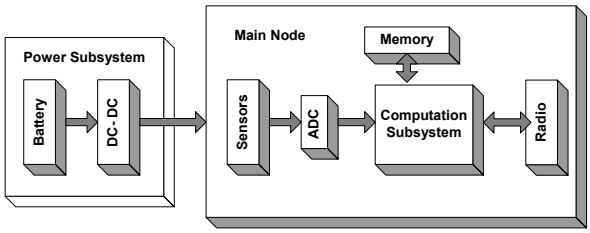


Figure 2 Typical sensor node architecture

Figure 2 depicts the Medusa MK-2 architecture. The computation subsystem of the node consists of two microcontroller. The first one is an 8-bit 4MHz ATMega128L MCU [1] from Atmel. This has 32KB of flash and 4KB of RAM and it is used as an interface to the sensors and for radio baseband processing. The second one is a 16/32-bit AT91FR4081 ARM THUMB processor [2] also from Atmel. This is a more powerful processor based on an ARM7TDMI core running at 40MHz. It has 136KB of RAM and 1MB of on-chip FLASH memory and comes in a compact 120-ball BGA package. The communication subsystem is made up of a TR1000 low power radio from RF Monolithics [7] and an RS-485 serial bus transceiver for wireline communication. The sensing subsystem is made up of a MEMs accelerometer (ADXL202E from Analog Devices) and a temperature sensor. The node also has a rich set of interfaces: 8 10-bit ADC inputs, serial ports (I^2C, RS-232, RS-485, SPI) and numerous general purpose I/O (GPIO) ports. An accessory board implements an ultrasonic ranging subsystem uses a set of 40KHz ultrasonic transducers (both transmitters and receivers). These are used in coordination with RF transmissions to measure inter-node distances for node localization. In addition to the sensors, the node also has two pushbuttons that serve as a user interface. These are used to trigger events and to execute different tests during experimentation. The node has two external connectors (see Figure 3). The first one has all the necessary connections for communicating with a PC to download and debug software. The wiring required for connecting the node to an external GPS module is also provided on this connector. The second connector has a set of ADC, GPIO and communication lines and it serves as an expansion slot for attaching add-on boards carrying different sensors. The description of each of the node subsystems is provided in the next section.

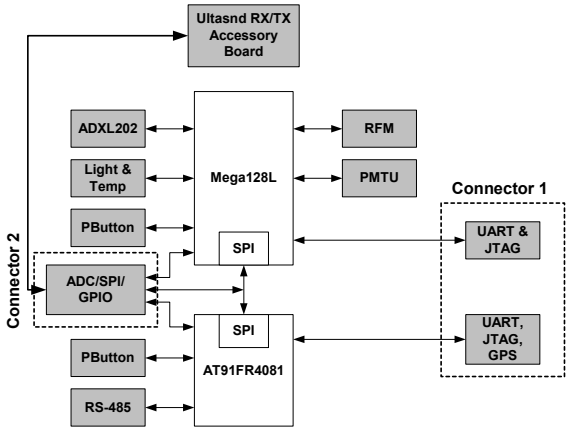


Figure 3 The Medusa MK-2 architecture

3. Medusa MK-2 Components

3.1 The Computation Subsystem

To design the computation subsystem, we classified the node computation tasks into two broad categories; low demand and high demand low frequency, according to their computation needs. The fist class contains the periodic tasks that the sensor node has to make such as the base band processing for the radio while listening for new packets, sensor sampling, handling of sensor events and power management. Although these tasks require a high degree of concurrency, they are not particularly demanding in terms of computation and can be easily handled an 8-bit microcontroller. The TinyOS [4] development effort at UC Berkeley has shown how such a task set can be supported by with a low power AVR microcontroller. The Medusa MK-2 architecture follows the same approach by dedicating an AVRMega128L microcontroller to handle these less computation demanding but highly concurrent tasks that a sensor node has to fulfill.

The second class of computation runs a set of algorithms that process acquired sensor data to produce a result or conclusion about what is being sensed. An example of such computation can be drawn from the fine-grained localization problem described in [5]. In this situation, a sensor node is expected to compute an estimate of its location by using a set of distance measurements to known landmarks or beacons. To solve this least squares estimation problem a node is required to perform a set of high precision matrix operations. This type of computation consumes between 3-4 MIPS [5] and has high accuracy requirements and it is more suitable for a higher end processor. Performing this computation on an 8-bit processor, would incur high latencies and less precision in the calculation due to round off errors. Instead the 32-bit instruction set and datapath provided by the 40MHz ARM THUMB processor is a more suitable environment for this type of computation. Furthermore, the THUMB microcontroller has sufficient resources to run some off the shelf embedded operating systems such as Red Hat eCos and uCLinux. This adds the additional advantage of allowing some of the existing applications and a rich set of libraries to run on the nodes.

This distribution of computation is also favorable from a power/latency perspective. The THUMB processor executes instructions at a rate of 0.9MIPS per MHz at 40MHz while drawing 25mA with a 3V supply. This gives a performance of 480 MIPS/Watt. The ATMega128L on the other hand operates at 4MHz and draws 5mA at a 3V supply thus provides 242 MIPS/Watt. Table 1 shows the microcontroller parameters which result in this power/latency tradeoff.

Table 1 MCU Comparison

	AT91FR4081	ATMega128L
Datapath	**16/32 Bit**	**8 bit**
Clock Speed (MHz)	**40**	**4**
MIPS/MHz	**(ARM 0.9), (THUMB 0.7)**	**1**
Power @ 3V(mW)	**75**	**15**
MIPS/W	**480**	**242**

The two processors communicate with each other with a pair of interrupt lines, one for each microcontroller, and an SPI bus. The microcontrollers use the interrupts as a mechanism for waking up each other from sleep mode when information exchange needs to take place. Information exchange takes place over SPI. The SPI interface was selected because of its high-speed capabilities (above 1Mbps). The SPI bus is included on connector 2 (see figure 3) so it can also support additional processors added to the node such as DSP processors or additional microcontrollers that are part of additional sensor boards.

3.2 The Communication Subsystem

The communication subsystem consists of both a wired and a wireless link. The wireless link is implemented with a low power TR1000 radio from RF Monolithics. This radio has a 0.75mW maximum transmit power and has an approximate transmission range of 20 meters. Additionally the radio supports two different modulation schemes, On-Off Keying (OOK) and Amplitude Shift Keying (ASK). The selection of a modulation scheme can be done in software according to the application specification. The radio supports multiple data rates ranging from 2.4kbps to 115kbps. On the Medusa MK-2 node, the base band processing for the radio is done by the ATMega128L microcontroller. This configuration allows running a lightweight medium access control (MAC) protocol on the ATMega128L processor. The S-MAC protocol presented in [12] is a low power MAC protocol for sensor networks that is well suited for this purpose.

In addition to the wireless front end, the Medusa MK-2 node is also equipped with an RS-485 serial bus interface for wireline communication. A low power RS-485 transceiver is attached to one of the RS-232 ports of the THUMB processor and allows the connecting the nodes to an RS-485 network using an RJ-11 connector and regular telephone wire. A single RS-485 network can have up to 32 nodes that can span over a total wire length distance of 1000 feet. Besides providing a wireless networking alternative in places with high interference where radios cannot function adequately this

configuration allows a wide variety of node configurations such as:

- **Array formations** – several nodes, each one equipped with different sensors can be daisy chained to form node arrays.
- **Gateway functions –** nodes can act as gateways, connecting other wireless nodes to the wired infrastructure. With the use of RS-485 several gateways can be attached to the same workstation.
- **Out-of-band data collection** – during experiments where the data is processed on the nodes and communicated over wireless links, the raw data can also be collected using the wired infrastructure for offline analysis later on.

3.3 The Power Subsystem

The power subsystem consists of 2 main units: the power supply and the Power Management and Tracking Unit PMTU [10]. The power supply consists of a 540mAh lithium-ion rechargeable battery and an up-down DC-DC converter that has a 3.3V output and can source up to 300mA of current from the battery. Although with no sensors attached, the node requires less than 50mA, the power supply designed to source up to 300mA currents to provide power-additional sensors than can be attached to the node as accessory boards. Table 2 shows the average current drawn by the main node components[1] during active and sleep node. According to the table, the maximum power consumption of the node is less than 150mW. During normal operation, the node consumes less power by putting the unused components in sleep mode. In a typical sensor network setting, the ARM THUMB processor together with the RS-485 and RS-232 transceivers are in sleep mode most of the time resulting up to an 80% reduction of the overall node power consumption.

Table 2 Current drawn by node components

Component	Active(mA)	Sleep(mA)
ATMega128L	5.5	1
RFM	2.9	5
AT91FR4081	25	10
RS-485	3	1
RS-232	3	10
Total	39.4	27

To get an indication of how the Medusa MK-2 power consumption relates to other sensor nodes, we compared its power consumption to the power consumption of a higher end node, the WINS node [8] developed at the Rockwell Science Center. This node is equipped with a more powerful StrongARM SA-1100 microprocessor from Intel, a 100-meter range 100Kbps radio from Connexant and several sensors. The results of the power characterization of the WINS node at different operational modes are shown in table 3. Table 4 shows the same characterization for the Medusa MK-2 node. Based on this comparison, the power consumption of the Medusa MK-2 node when all subsystems are active is approximately 10 times less than the power consumption of the WINS node. Furthermore, by shutting down the THUMB processor on the Medusa MK-2 node when not in use can result in 44 times less power consumption than the WINS node.

Table 3 Power Characterization of WINS node

MCU Mode	Sensor Mode	Radio Mode	Power(mW)
Active	On	Tx(Power:36.3mW)	1080.5
		Tx(Power:19.1mW)	986.0
		Tx(Power:13.8mW)	942.6
		Tx(Power:3.47mW)	815.5
		Tx(Power:2.51mW)	807.5
		Tx(Power:0.96mW)	787.5
		Tx(Power:0.30mW)	773.9
		Tx(Power:0.12mW)	771.1
		Rx	751.6
		Idle	727.5
		Sleep	416.3
		Removed	383.3
Sleep		Removed	64.0
Active	Removed	Removed	360.0

Another interesting observation noted in the power measurements is that the power consumption of the radio is almost the same regardless whether the radio is in receive transmit or idle mode. This implies that no power is conserved when the radio is in idle state, so it is better to develop protocols that completely shutoff the radio when not in use, hence a media access control protocol like S-MAC is highly desirable.

Table 4 Power characterization of Medusa MK-2 node

AVR MCU Mode	Sensor Mode	Radio Mode	Modulation	Data Rate(kbps)	Power (mW)	
					THUMB OFF	THUMB ON
Active	On	Tx (Power: 0.7388 mW)	OOK	2.4	24.58	107.08
		Tx (Power: 0. 0.979mW	OOK	2.4	19.24	101.74
		Tx (Power: 0.7388 mW)	OOK	19.2	25.37	107.87
		Tx (Power: 0. 0.979mW	OOK	19.2	20.05	102.55
		Tx (Power: 0.7388 mW)	ASK	2.4	26.55	109.05
		Tx (Power: 0. 0.979mW	ASK	2.4	21.26	103.76
		Tx (Power: 0.7388 mW)	ASK	19.2	27.46	109.96
		Tx (Power: 0. 0.979mW	ASK	19.2	22.06	104.56
Active	On	Rx	Any	Any	22.20	104.7
Active	On	Idle	Any	Any	22.06	104.56
Active	On	Off	Any	Any	9.72	92.22
Idle	On	Off	Any	Any	5.92	88.42
Sleep	Off	Off	Any	Any	0.02	82.52

To further reduce power consumption, the Medusa MK-2 node is equipped with a power Management/Tracking Unit (PMTU). This is a set of three DS2438 battery monitors from Dallas Semiconductor that keep track of the power consumed by the different node sub-systems. The first battery monitor keeps track of the power consumed by the AT91FR4081 processor, the second tracks the power consumed by the radio while the third monitors the overall node power consumption. Using the PMTU information, the Medusa MK-2 node can implement power aware algorithms to maximize battery

[1] Numbers obtained from data sheets

lifetime. By making this power consumption information available to the application level, applications can set up their own power aware policies and decide which parts of the node to shutdown in order to conserve energy while meeting their sensing, computation and communication requirements.

4. An example application: Node Localization

To illustrate the use of the Medusa MK-2 node as a distributed computation and sensing platform we use an instantiation of the multihop node localization problem described in [5]. In this problem, nodes with unknown locations (white nodes in Figure 4) are expected to estimate their locations by setting up and solving a global non-linear optimization problem. To solve this problem, nodes first "sense" their separation to their neighbors using the node's ultrasonic ranging subsystem. When all the required measurements are made, the nodes with unknown positions combine these measurements with known location information of landmark nodes (black nodes in Figure 4) to estimate their locations using *distributed collaborative multilateration*.

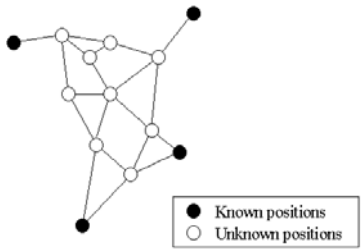


Figure 4 Solving for node positions in a multihop network

In this type of setup, the optimal position estimate is the one computed from a global vantage point that considers all the physical topology constraints. This however is a large non-linear optimization problem that computation and memory-constrained nodes cannot solve individually. With distributed collaborative multilateration nodes with unknown locations in setups similar to the one in figure 4 are able to estimate their locations locally while taking global constraints into consideration. As it was shown in [5], using this fully distributed computation model, resource constrained MK-2 nodes with unknown position can collaborate with each other to estimate their physical positions, a task that none of the nodes can perform individually.

5. Related Work

Research efforts in the last few years have produced a wide variety of sensor nodes ranging from tiny sensor nodes promised by the Smart Dust project [6] to fully-fledged nodes such as the WINS nodes [11] produced by Sensoria Corporation. The Smart Dust nodes still in development promise cubic millimeter scale form factor and a few cents per node manufacturing cost. The WINS nodes are already in use by the research community. They feature a Hitachi SH4 floating-point processor running linux and a long-range frequency hopping radio. Although these nodes are very powerful for some applications they are still large and power hungry and fairly expensive for some indoor applications and building large experimental networks in a lab setting.

UC Berkeley's MICA nodes [13] are an example of lower cost nodes that is currently widely used within the research community. The MK-2 node shares many similarities with this node. It uses the same AVR microcontroller and radio, it can support similar sensors and it is interoperable with the Mica modes. MK-2 differs from the Mica motes in that it has additional processing power, larger power supply and a set of customized features and sensor interfaces geared towards experimentation, especially for node localization problems.

8. Conclusions

We have presented the Medusa MK-2 node, a wireless node for distributed computation and sensing. The main focus of our development is to produce a simple, low cost design that is easy to program and provides great flexibility for experimentation in many different settings. We believe that this node will provide an affordable solution for constructing reasonable sized testbeds that would help in the development and validation of new protocols and concepts in this new era of wireless embedded sensing.

Acknowledgements

This paper is based in part on research funded through NSF under grant number ANI-008577, and through DARPA SensIT and Rome Laboratory, Air Force Material Command, USAF, under agreement number F30602-99-1-0529. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF, DARPA, or Rome Laboratory, USAF.

References:

[1] ATMega 128L Datasheet, Atmel Corporation
http://www.atmel.com/atmel/acrobat/doc0945.pdf

[2] AT91FR4081 Datasheet, Atmel Corporation, http://www.atmel.com/atmel/acrobat/doc1386.pdf

[3] M. B. Srivastava, R. Muntz and M. Potkonjak, *Smart Kindergarten: Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments*, Proceedings of the ACM SIGMOBILE 7th Annual International Conference on Mobile Computing and Networking, Rome, Italy, July 2001

[4] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, *System Architecture Directions for Network Sensors,* Proceedings of ASPLOS 2000.

[5] A. Savvides, H. Park and M. B. Srivastava, *The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems*, NESL Technical Report TM-UCLA-NESL-2002-03-07, March 2002, available from http://nesl.ee.ucla.edu/projects/ahlos/reports.htm

[6] J.M Kahn, R. H. Katz and K. S.J. Pister, *Next Century Challenges: Mobile Networking for Smart Dust*, in proceedings of Mobicom 99, pp 483-492

[7] TR1000 Radio Module, RF Monolithics, http://www.rfm.com/products/data/tr1000.pdf

[8] Wireless Integrated Network Systems (WINS), http://wins.rsc.rockwell.com

[9] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, *Next Century Challenges: Scalable Coordination in Sensor Networks*, Proceedings of the fifth annual international conference on Mobile Computing and Networking, Seattle, Washington, 1999, Pages 263-270

[10] A. Chen, R. Muntz, S. Yuen, I. Locher, S. Park and Mani B. Srivastava, *A Support Infrastructure for the Smart Kindergarten*, IEEE Pervasive Computing Magazine, vol 1, number 2, April-June 2002 pp. 49-57

[11] G. J. Pottie and W. J. Kaiser, *Wireless Intergrated Network Sensors*, Communications of the ACM, vol. 43, no. 5, pp. 51-8, May 2000

[12] Wei Ye, John Heidemann and Deborah Estrin, *An Energy Efficient MAC Protocol for Sensor Networks* Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), New York, NY, USA, June, 2002.

[13] Mica Motes, Crossbow, http://www.xbow.com/Products/Wireless_Sensor_Networks.htm

Appendix P:

Design and Implementation of a Framework for Efficient and Programmable Sensor Networks

Abstract – **Wireless ad hoc sensor networks have emerged as one of the key growth areas for wireless networking and computing technologies. So far these networks/systems have been designed with static and custom architectures for specific tasks, thus providing inflexible operation and interaction capabilities. Our vision is to create sensor networks that are open to multiple transient users with dynamic needs. Working towards this vision, we propose a framework to define and support lightweight and mobile control scripts that allow the computation, communication, and sensing resources at the sensor nodes to be efficiently harnessed in an application-specific fashion. The replication/migration of such scripts in several sensor nodes allows the dynamic deployment of distributed algorithms into the network. Our framework, SensorWare, defines, creates, dynamically deploys, and supports such scripts. Our implementation of SensorWare occupies less than 180Kbytes of code memory and thus easily fits into several sensor node platforms. Extensive delay measurements on our iPAQ-based prototype sensor node platform reveal the small overhead of SensorWare to the algorithms (less than 0.3msec in most high-level operations). In return the programmer of the sensor network receives compactness of code, abstraction services for all of the node's modules, and in-built multi-user support. SensorWare with its features apart from making dynamic programming possible it also makes it easy and efficient without restricting the expressiveness of the algorithms.**

I. INTRODUCTION

Wireless ad-hoc sensor networks (WASNs) have drawn a lot of attention in recent years from a diverse set of research communities. Researchers have been mostly concerned with exploring applications such as target tracking and distributed estimation, investigating new routing and access control protocols, proposing new energy-saving algorithmic techniques for these systems, and developing hardware prototypes of sensor nodes.

Little concern has been given on how to actually program the WASN. Most of the time, it is assumed that the proposed algorithms are hard-coded into the memory of each node. In some platforms the application developer can use a node-level OS (e.g. TinyOS) to create the application, which has the advantages of modularity, multi-tasking, and a hardware abstraction layer. Nevertheless the developer still has to create a single executable image to be downloaded manually into each node. However, it is widely accepted that WASNs will have long-deployment cycles and serve multiple transient users with dynamic needs. These two features clearly point in the direction of dynamic WASN programming.

What kind of dynamic programmability do we want for WASNs? Having a few algorithms hard-coded into each node but tunable through the transmission of parameters, is not flexible enough for the wide variety of possible WASN applications. Having the ability to download executable images into the nodes is not feasible because most of the nodes will be physically unreachable or reachable at a very high cost. Having the ability to use the network in order to transfer the executable images to each and every node is energy inefficient (because of the high communication costs and limited node energy) and cannot allow the sharing of the WASN by multiple users. What we ideally want is to be able to dynamically program the WASN as *a whole, an aggregate*, not just as a mere collection of individual nodes. This means that a user, connected to the network at any point, will be able to inject instructions into the network to perform a given (possibly distributed) task. The instructions will task individual nodes according to user needs, network state, and physical phenomena, *without any intervention from the user*, other than the initial injection. Furthermore, since we want multiple users to use the WASN concurrently, several resources/services of the sensor node should be abstracted and made sharable by many users/applications.

One approach of programming the WASN as an aggregate is a distributed database system (e.g., [1]). Multiple users can inject database-like queries to be autonomously distributed into the network. The WASN is viewed as a distributed database and the query's task is to retrieve the needed information by finding the right nodes and possibly aggregate the data as they are routed back to the user. This approach ignores though the fact that information is not always resident in nodes but sometimes has to be retrieved by *custom collaboration* among a changing set of nodes (e.g., target tracking). Thus even though the database model is programming the network in the desirable way, it is not expressive enough to implement any distributed algorithm.

The other approach to WASN programmability that is used by our framework, and is gaining momentum lately,

is the "active sensor" approach. This term was used in [19], to describe a family of frameworks that try to task sensor nodes in a custom fashion, much like active networking frameworks task data network nodes. The difference is that while active networking tasks are reacting only to reception of data packets, active sensor tasks need to react to many types of events, such as network events, sensing events, and timeouts. Active sensor frameworks abstract the run-time environment of the sensor node by installing a virtual machine or a high-level script interpreter at each node. For example, single instructions of the scripts (or bytecodes) can send packets, or read data from the sensing device. Moreover, the scripts (or bytecodes) are made mobile through special instructions, so nodes can autonomously task their peers.

The difficulty in designing an active sensor framework is how to properly define the abstraction of the run-time environment so that one achieves compactness of code, sharability of resources for multi-user support, portability in many platforms, while at the same time keeping a low overhead in delays and energy. Our proposal of such a framework, called SensorWare, employs lightweight and mobile control scripts that are autonomously populated in sensor nodes after a triggering user injection. The sensor node abstraction was made in such a way so that multi-user accessibility is given to all of the node's modules (e.g., radio, sensing devices) while also creating other services (e.g., real-time timers). Considerable attention was given to the portability and expandability of the framework by allowing the definition of new modules. By choosing the right level of abstraction the scripts are compacted to 10s-100s of bytes. For the non-trivial application examined in section V.A, the SensorWare script is smaller than the code of other frameworks with comparable capabilities in algorithm expressiveness (e.g. other active sensors scripts, binary images).

Our implementation and porting of SensorWare in several sensor node platforms shows that the size of the framework is small enough (<180Kbytes) to fit in most current sensor node designs. Moreover, extensive measurements in our prototype iPAQ-based sensor node platform reveal the delay and energy overheads of SensorWare. Every SensorWare script command has a delay less than 0.3msec showing the limits of real-time operation. Note that the script commands have a high-level of abstraction (i.e., each command performs multiple low-level operations). Experiments with both compiled and interpreted versions of the scripts are conducted in order to explore the energy trade-off space between different representations of an algorithm.

Section II discusses in depth the nature of WASNs, approaches to WASN programmability, and the general idea of our approach. Section III presents related work. Section IV presents SensorWare's architecture. Section V illustrates how is SensorWare ported to a platform and explains a moderately large script solving a real problem. Section VI presents our current implementation and the measurements we acquired through it. Finally, section VII concludes the paper.

II. Motivation and Background

A. *Wireless Ad hoc Sensor Networks*

Figure 1 shows an example of a WASN, highlighting its main characteristics. An ad hoc network of miniature, resource-limited, static, wireless, sensor nodes is being used to monitor a dynamic physical environment. The use of low power communication and the need for diversity in sensing necessitates a multi-hop, distributed architecture [22]. The computation capabilities at the nodes can be leveraged for event detection via data fusion and collaborative signal processing among nearby nodes, so that higher bandwidth raw sensor data does not need to be sent to the users. Typically a user queries the network (consider the term "query" in the broad sense, not just database query), the query triggers some reaction from the network, and as the result of this reaction the user receives the information needed. The reaction to the query can vary from a simple return of a sensor value, to a complex unfolding of a distributed algorithm among some or all of the sensor nodes, such as a collaborative signal processing algorithm or a distributed estimation algorithm. Furthermore, there are multiple users who are transiently connected to the network, each having different needs in requested information. The WASN is there to accommodate all or most of their needs.

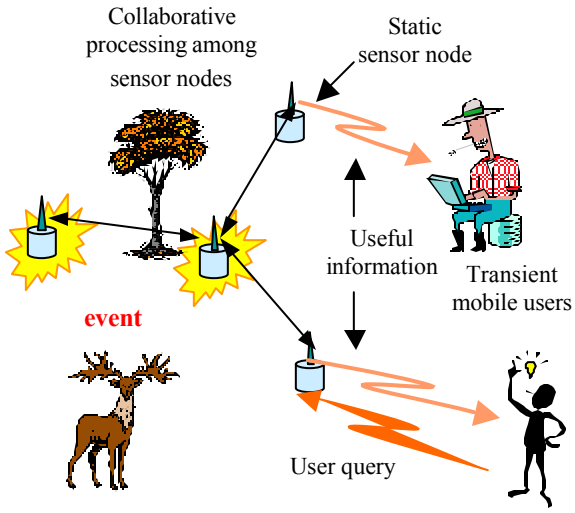


Figure 1: Wireless Ad-hoc Sensor Network

These systems are quite different from traditional networks. First, they have severe energy, computation, storage, and bandwidth constraints. Second, their overall usage scenario and the implications that this brings to the traffic and the interaction with the users is quite different from traditional networks. There is not a mere exchange of data between users and nodes. The user will rarely be

interested in the readings of one or two specific nodes. The user will be interested in some parameters of a dynamic physical process. To efficiently achieve this, the nodes have to form an application-specific distributed system to provide the user with the answer. Moreover, the nodes that are involved in the process of providing the user with information are constantly changing as the physical phenomenon is changing. Therefore the user interacts with the system as a whole. The WASN is not there to connect different parties together as in the traditional networking sense but to provide information services to users.

As a consequence efficiently designed WASNs operate in a fashion where a node's actions are affected largely by physical stimuli detected by the node itself or nearby nodes. Frequent long trips to the user are undesirable because they are time and energy consuming. This decentralized (i.e. not all traffic flows to/from user), autonomous (i.e., user out-of-the-loop most of the time) way of operating, is called "proactive computing" (as opposed to interactive) by David Tennenhouse [27]. We also adopt the term "proactive" throughout the paper to denote an autonomous and non-interactive nature.

Efficiently designed WASNs are application-specific distributed systems that require a different distributed proactive algorithm as an efficient solution to each different application problem. Given the nature of SNs, one can coarsely define two classes of problems in their design. First, the application-specific problem: How does one find the most efficient distributed algorithm for a particular problem? Second, the generic problem: How does one dynamically deploy different algorithms into the network, what is the programming model that will implement these algorithms, and what general support does one need from the framework?

For the first class of problems (i.e., finding efficient algorithms for particular applications), there are many research efforts in a variety of application problems (e.g., target tracking, sensor reading aggregation). In this paper we will not expand into any particular application problem. We only note, that in general, localized distributed algorithms (i.e., distributed algorithms that act locally, using only local information) are particularly efficient in most WASN problems as they achieve small and well-distributed energy consumption, thus prolonging the network lifetime.

The second class of problems (i.e., what is the right framework to express and dynamically deploy distributed algorithms for WASNs) is the focus in this paper. We describe our proposal of such a framework, called SensorWare. SensorWare provides a language model powerful enough to express the most efficient distributed algorithms while at the same time hiding unnecessary low-level details from the application programmer and providing a way to share the resources of a node among many applications and users that might concurrently use the WASN. The language model is developed after examining what are the properties of efficient algorithms for SN (e.g., localized distributed algorithms), and in conjunction with developing our own applications on real sensor networks [3].

Equally important is the role of SensorWare in the dynamic deployment of the distributed algorithms into the network. As sensor nodes are memory-constrained, they cannot store every possible application in their local memory. Thus, a way of dynamically deploying a new application is needed. Usually this means that a distributed algorithm has to be incorporated in several sensor nodes, which in turn means that these sensor nodes have to be dynamically programmed. A user-friendly and energy-efficient way of programming the nodes keeps the user out-of-the-loop most of the time by allowing sensor nodes to program their peers. By doing so, the user does not have to worry about the specifics of the distributed algorithm (because the information on how the algorithm unfolds lies within the algorithm), and the nodes save communication energy (because they interact with their immediate neighbors and not with the user node through multi-hop routes). The programming model of SensorWare is designed in such a way, so as to facilitate the user-friendly and energy-efficient dynamic deployment of an algorithm. The user "injects" the query/program into the network, and the query *autonomously* unfolds the distributed algorithm into the nodes that should be affected.

B. Approaches to WASN programmability

As discussed in the introduction, one of the approaches currently under investigation is a distributed database model. A good example of this approach is the work done at Cornell [1]. A similar scheme called DataSpace focusing on location addressing has also been developed in Rutgers [14]. Each node is equipped with a fixed database query resolver. As queries arrive to a node, the local resolver decides on the best, distributed plan to execute the query and distributes the query to the appropriate nodes. Although this approach takes into account the distributed nature of the system and works well in several scenarios, it does not take into account the proactive nature of the system. The user is the central place of control and most data flows to/from the user. This property can prove inefficient in applications such as target tracking, where it is better for nodes to form clusters around the target, collaboratively compute the target's location and just send the location information back to the user. Clearly a more flexible way of programming the sensor network is needed to enable this kind of behavior.

C. SensorWare

The SensorWare architecture is based on a scriptable lightweight run-time environment, optimized for sensor nodes that have limited energy and memory. This environment securely hosts one or more simple, compact, and platform-independent sensor-node control scripts. The sensing, communication, and signal-processing resources

of a node are exposed to the control scripts that orchestrate the dataflow to assemble custom protocol and signal processing stacks. SensorWare has to also promote the creation of distributed proactive algorithms based on the scripting language described above. For this reason the scripts are made mobile using special language commands and directives. A script can replicate or migrate its code and data to other nodes, directly affecting their behavior. The replication or migration of a script will be called "population" in the paper.

A usage scenario can be like the following: A user sends a query to the sensor network. The query is a script, a state machine in its simplest form, which is injected to one or more sensor nodes. The script will describe among other things how it is going to populate itself to other nodes. The process of population can continue depending on events and the current state. For example as the events of interest are moving to a different area, the scripts can move along with them, possibly trying to predict their next move. The populated scripts will collaborate among themselves in order to extract the information needed by the user, and when this information is acquired it is sent back to the user.

III. Related Work

SensorWare falls under the family of active sensor frameworks. Its closest relatives in the traditional networks realm are Mobile Agent frameworks. Other active networking frameworks exhibit similarities, such as the scripting abstraction. In this section we only consider work that tries to make WASNs programmable using active sensor concepts. Therefore, general mobile-agent and active-network platforms are not discussed, nor any distributed database systems for WASNs are presented. The interested reader can refer to [2] for a comprehensive comparison of SensorWare with mobile agent platforms, as well as with an active networking framework called PLAN [10].

An active sensor framework for WASNs is currently being developed in Berkeley under the name Maté. Maté [19] is a tiny virtual machine build on top of TinyOS [12]. TinyOS is an operating system, designed specifically for the Berkeley-designed family of sensor nodes, generically named "motes" [11][12]. Maté's goal is to make a WASN made of motes dynamically programmable in an efficient manner. This includes the capability to dynamically instruct a mote to execute any program, and expressing this program in a concise way. They achieve this by building a virtual machine (VM) for the motes. The virtual machine supports a very simple, assembly-like language, to be used for all needs of mote-tasking. Programs (called capsules) written on the VM language can be injected to any node and perform a task. Furthermore the capsules have the ability to self-transfer themselves by using special language commands. This model seems extremely like our own in SensorWare. Indeed, Maté shares the same goals as SensorWare as well as the same basic principles to achieve these goals. Differences appear though when one looks thoroughly into each platform's implementation.

Maté, like its substrate TinyOS, was built with a specific platform in mind: the extremely resource-limited mote. The main restriction for the developer of mote-targeted frameworks (such as an OS or a VM) is memory. The newest version of a mote called mica offers 128Kbytes of program memory and 4Kbytes of RAM. An older version called rene2 has 16Kbytes of program memory and 1Kbyte of RAM. Maté, with an ingenious architecture, supports both platforms. Being so memory constrained, Maté has to sacrifice some features that would make programming easier and more efficient. First, a stack-based architecture with an ultra-compact instruction set (all instructions are 1 byte) is adopted which is reminiscent of a low-level assembly language or the byte code of the Java VM. This kind of model makes programming of even medium-sized tasks difficult. Furthermore, due to the ultra-compact instruction set, many 1-byte instructions are needed to express a medium complexity algorithm, which in turn leads to large programs, compared to a higher-level, more abstracted scripting language. The size of programs is important since the code is transmitted/received using the radios of the nodes spending energy for every transmitted/received bit. Second, the behavior of a program when radio packets are received is rather rigid. A handler to process such events is essentially stateless in Maté. Thus, if a new pattern of packet processing is needed, a new handler has to be transferred through the network. This imposes an overhead in energy consumption and execution time. Third, because there is only one context (i.e., handler) per event (e.g., clock tick, reception of packet) multiple applications cannot run concurrently in one mote.

SensorWare cannot fit in the restricted memory of a mote. SensorWare targets richer platforms that we believe are going to be the mainstream in sensor node design in the immediate future. Such platforms (e.g., [24]) include a 1Mbyte of program memory and 128Kbytes of RAM. Having the luxury of more memory, SensorWare supports easy programming with a high-level scripting language, as well as concurrent multi-tasking of a node so that multiple applications can concurrently execute in a WASN. The programming model and properties of SensorWare are extensively discussed in section IV.

Particularly instructive is to study the relationship between SensorWare's mobile scripting approach and the mobile code approach in Penn State's Reactive Sensor Network [23] (RSN) project under DARPA's SenseIT program [25]. RSN's focus is on providing an architecture whereby sensor nodes can: (i) download executables and DLLs, identified by URLs, from repositories or their cache, (ii) execute the program at the local node using input data which itself may be remotely located and identified by a URL, and (iii) write the data to a possibly remote URL. The RSN model is in essence Java's applet

model generalized to arbitrary executables and data, and combined with a lookup service. The focus of RSN is quite different from SensorWare. Differences include: (i) RSN provides a general lookup and download service, (ii) RSN does not seek to provide a scripting environment with an associated sensor node resource model for use by scripts, and (iii) RSN's notion of mobility is download oriented, as opposed to SensorWare's approach of a script which can autonomously spawn scripts to remote nodes. RSN views sensor nodes as network switches with dynamically adaptable protocols, trying to directly map the motivation and methods of classical active networks into sensor networks. Unfortunately such an approach does not address the basic problems of sensor networks. Although one might be able to construct some distributed applications using the above scheme, by no means the creation and diffusion of distributed proactive applications into the network is supported by its architecture.

Finally, extremely relevant is the work that is being conducted in University of Delaware by Jaikaeo et al. [16] called SQTL (Sensor Querying and Tasking Language). Having the same goals as our research, but starting from a different point (database-like queries), the researchers end up with the same basic solution as SensorWare, namely a tasking language for sensor networks. To lively demonstrate the relevance to our work we are quoting an excerpt from [16]. *"We model a sensor network as a set of collaborating nodes that carry out querying and tasking programmed in SQTL. A frontend node injects a message, that encapsulates an SQTL program, into a sensor node and starts a diffusion computation. A sensor node may diffuse the encapsulated SQTL program to other nodes as dictated by its logic and collaboratively perform the specified querying or tasking activity."*

SQTL fits in a more general architecture for sensor networks called SINA (Sensor Information Networking Architecture) [26]. SINA uses both SQL-like queries as well as SQTL programs. Some of its main features include: 1) hierarchical clustering, 2) attribute-based naming, 3) a spreadsheet paradigm for organizing sensor data in the nodes. SQL-like queries use these three features to execute simple querying and monitoring tasks. When a more advanced operation is needed though, SQTL plays the essential role by programming (or "tasking" as the researchers from Delaware call it) the sensor nodes and allowing proactive population of the program. In SINA, SQTL is used as an enhancement of simple SQL-like queries. The framework is there mainly to support the queries not the mobile scripts. As a consequence, SQTL scripts do not have all the provisions that SensorWare scripts have. The most important of them are: 1) Rich sensor-node-related APIs (e.g. for networking, sensing). 2) Diverse rules for mobility. A SQTL script can only specify the nodes to be populated. SensorWare first checks if the script is already in the remote node and offers a multitude of possibilities depending on how many instances of the script are already running in the remote node. 3) Code modularity in order to share functionality and avoid redundant code transfers 4) Support for multi-user scripts. 5) Resource management in the presence of multiple scripts running in the node.

IV. ARCHITECTURE

First, we show SensorWare's place inside the overall sensor node's architecture (Figure 2). The architecture of a sensor node can be viewed in layers. The lower layers are the raw hardware and the hardware abstraction layer (i.e., the device drivers). An operating system (OS) is on top of the lower layers. The OS provides all the standard functions and services of a multi-threaded environment that are needed by the layers above it. The SensorWare layer for instance, uses those functions and services offered by the OS to provide the run-time environment for the control scripts. The control scripts rely completely on the SensorWare layer while populating around the network. Static applications and services coexist with mobile scripts. They can use some of the functionality of SensorWare as well as standard functions and services of the OS. These applications can be solutions to generic sensor node problems (e.g., location discovery), and can be distributed but not mobile. They will be part of the node's firmware.

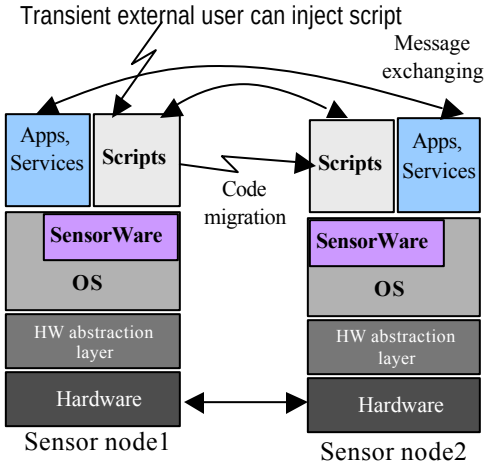


Figure 2: The general sensor node architecture

Two things comprise SensorWare: 1) the language, and 2) the supporting run-time environment. The next two subsections describe each of the parts in detail. A third subsection discusses issues of portability and expandability, and presents the final SensorWare code structure.

A. *The language*

As discussed earlier, the basic idea is to make the nodes programmable through mobile control scripts. Here the basic parts that comprise the language will be described as well as the programming model that emerges from the parts.

First, a scripting language needs proper functions/commands to be defined and implemented in order to use them as building blocks (i.e., these will be the basic commands of the scripts). Each of these commands will abstract a specific task of the sensor node, such as communication with other nodes, or acquisition of sensing data. These commands can also introduce needed functionality like moving a script to another node or filtering the sensing data through a filter implemented in native code. Second, a scripting language needs constructs in order to tie these building blocks together in control scripts. Some examples include: constructs for flow control, like loops and conditional statements, constructs for variable handling and constructs for expression evaluation. We call all these constructs the "glue core" of the language, as they combine several of the basic building blocks to make actual control scripts.

Figure 3 illustrates the different parts of the SensorWare language. Several of the basic commands/functions are grouped in theme-related APIs. We use the term API in a generic fashion, to denote a collection of theme-related functions that provide a programming interface to a resource or a service. As the figure hints, there is a question on what happens when we are dealing with different sensor node platforms that may support different/additional kinds of modules. Do we allow the set of APIs to be expandable? If so, who has the authority to name and define new commands? We will return to this topic with a solution in subsection C.

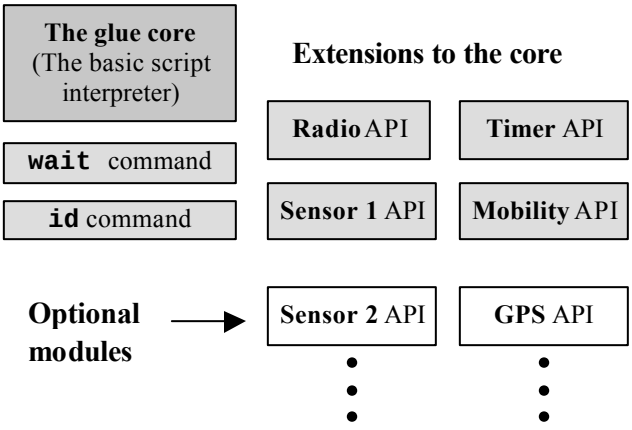


Figure 3: The language parts in SensorWare

As a glue core we can use the core from one of the scripting languages that are freely available, so we are not burdened with the task of building and verifying a core. One such scripting language, that is well suited for SensorWare's purposes, is Tcl [20], offering great modularity and portability. Thus, the Tcl core is used as the glue core in the SensorWare language. All the basic commands, such as `wait`, or the ones included in the APIs, are defined as new Tcl commands using the standard method that Tcl provides for that purpose.

The set of APIs is basically a way of easily exporting services and shared resources to the scripts. For example, the Timer API defines and sets/resets real time timers, while the Mobility API provides the basic functions to the scripts so they can transfer themselves around the network.

A.1 The general programing model

As discussed earlier, according to the proactive distributed model the scripts will look mostly like state machines that are influenced by external events. Such events include network messages from peers, sensing data, and expiration of timers. The programming model that is adopted is equivalent to the following: An event is described, and it is tied with the definition of an event handler. The event handler, according to the current state, will do some (light) processing and possibly create some new events or/and alter the current state. Figure 4 illustrates SensorWare's programming model with an example.

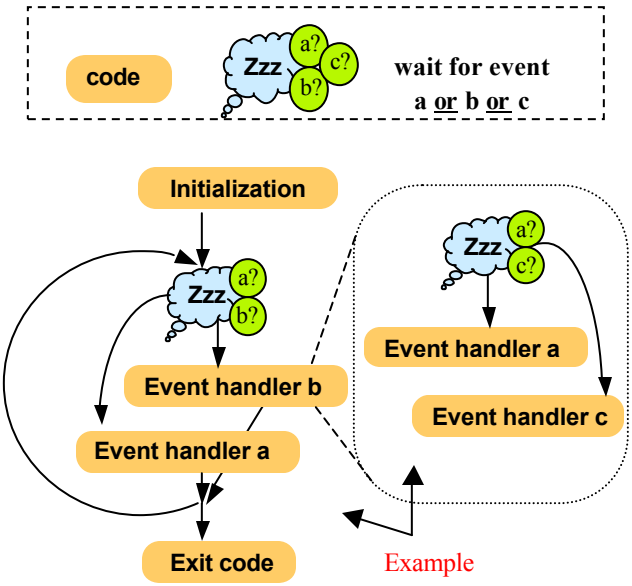


Figure 4: The programming model

The behavior described above is achieved through the `wait` command. Using this command, the programmer can define all the events that the script is waiting upon, at a given time. Examples of events that a script can wait upon are: i) reception of a message of a given format, ii) traversal of a threshold for a given sensing device reading, iii) filling of a buffer with sensing data of a given sampling rate, iv) expiration of several timers. When *one* of the events declared in the `wait` command occurs, the command terminates, returning the event that caused the termination. The code after the `wait` command processes the return value and invokes the code that implements the proper event handler. After the execution of the event handler, the script moves to a new `wait` command, or more usually it loops around and waits for events from the same `wait` command.

B. The run-time environment

As important are the scripts in the SensorWare platform, equally important is the run-time environment that supports them. Figure 5 illustrates the basic tasks performed by the environment. We separate tasks into fixed and platform-specific. The fixed tasks are always included in a SensorWare implementation, while the platform-specific depend on the existence of specific modules and services in the node platform. Again, the problem of expandability and portability appears. Do we allow any developer to *arbitrarily* define and create any tasks, according to the specific needs of each platform? Subsection C addresses this question.

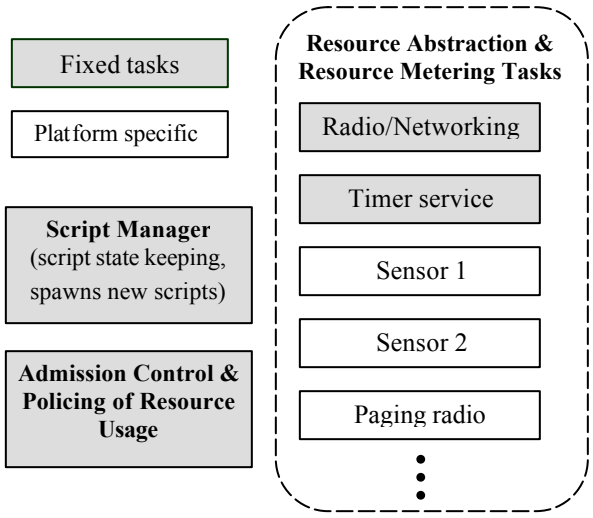
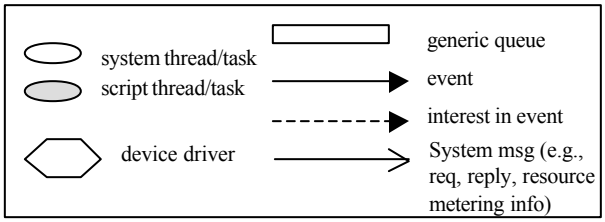


Figure 5: Tasks in the SensorWare run-time environment

The Script Manager is the task that accepts all requests for the spawning of new scripts. It forwards the request to the Admission Control task and upon receiving a positive reply, it *initiates a new thread/task running a script interpreter for the new script*. The Script Manager also keeps any script-related state such as script-data for as long as the script is active. Possible attacks, such as snooping or spoofing, are banned by the strict security model. The script manager also keeps a script-code cache in order to reduce code transmissions over the wireless channel. The Admission Control and Policing of Resource Usage task, as the name reveals, takes all the script admission decisions, makes sure that the scripts stay under their resource contract, and most importantly checks the overall energy consumption. If the overall consumption exhibits alarming characteristics (e.g., the current rate cannot support all scripts to completion) the task selectively terminates some scripts according to certain SensorWare policies. Resource management and the security model are not discussed in this paper. The interested reader can refer to [2].

The run-time environment also includes "Resource Abstraction and Resource Metering" tasks (sometimes referred to as "Resources Handling" tasks for brevity). Each task supports the commands of the corresponding APIs and manages a specific resource. There are two fixed tasks in this category since every platform is assumed to have at least one radio and a timer service. The "Radio" task manages the radio: i) it accepts requests from the scripts about the format of network messages that they expect, i) it accepts all network messages and dispenses them to the appropriate scripts according to their needs, and finally iii) measures the radio utilization for each script, a quantity that is needed by the "Admission Control & Policing of Resource Usage" task. The second fixed task, the "Timer service", accepts the various requests for timers by all the scripts and manages to service them using a real-time timer the embedded system provides. In essence the task provides many virtual timers relying on one timer provided by the system. According to platform capabilities a specific porting of SensorWare may run additional tasks For instance, a "Sensor Abstraction" task manages a sensing device. It accepts all requests for sensor data from all the scripts and decides on the optimal way to control the sensing device (e.g., setting the A/D sampling rate). It also measures the sensing device utilization for each script. Figure 6 depicts an abstracted view of SensorWare's run-time environment for an example platform with one sensing device.



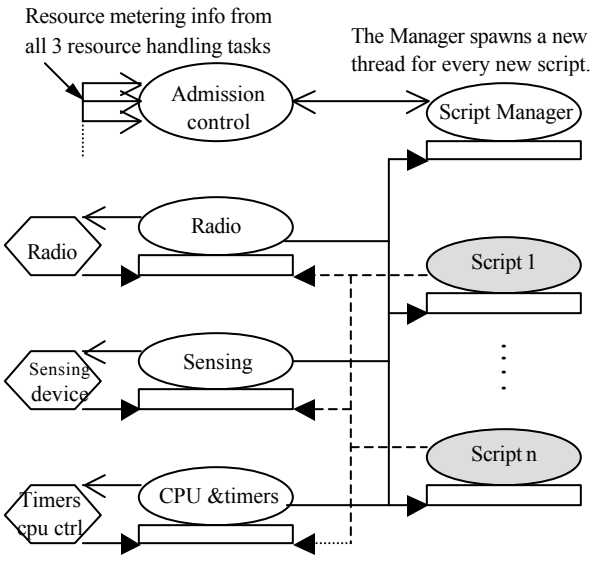


Figure 6: Abstracted view of SensorWare's run-time environment for an example platform

Most of the threads running are coupled with a generic queue. Each thread "pends" on its corresponding queue, until it receives a message in the queue. When a message

arrives it is promptly processed. Then the next message will be fetched, or if the queue is empty, the thread "pends" again on the queue. A queue associated with a script thread is receiving events (e.g., reception of network messages, sensing data, or expiration of timers). A queue associated with one of the resource handling tasks, receives events of one type (from the specific device driver that is connected to), as well as messages that declare interest in this event type. For instance, the Sensing resource-handling task is receiving sensing data from the device driver and interests on sensing data from the scripts. The Script Manager queue receives messages from the network that wish to spawn a new script. There are also system messages that are exchanged between the system threads (like the ones that provide the Admission Control thread with resource metering information, or the ones that control the device drivers).

C. *Portability and expandability of SensorWare*

In the previous subsections the problem of platform variability was revealed. Here we will present a solution for SensorWare's code structure. There are two kinds of platform variability: 1) capabilities variability (i.e. having different modules, such as sensing devices, GPS), 2) HW/SW variability (i.e. although the capabilities are the same we have different OS and/or specifics of hardware devices). We will explore solutions for each kind in two different subsections.

C.1 Capabilities variability

Different platforms may have different capabilities. For instance, imagine that one platform A has a radio and a magnetometer, while another platform B has two radios (a normal and a paging one) and a camera. How will we abstract the two platforms with the same framework? Since SensorWare's building blocks are the interface to the abstracted modules/services, we can allow an expandable API. Further, most modules/services will need a supporting task (as described in subsection B), so we can allow the definition and addition of arbitrary tasks in SensorWare's run-time environment. This kind of solution would create severe problems in the manageability of the code by different developers. SensorWare advocates a more modular and well-structured solution. SensorWare declares, defines, and support virtual devices (an idea triggered by Linux's virtual devices). Any module or service is represented as a virtual device. For example a radio, a sensing device, the timer service, a location discovery protocol are all view as virtual devices.

There is a **fixed interface** for all devices. More specifically there are four commands that are used to communicate with the device. They are: `query`, `act`, `createEventID`, and `disposeEventID`. `Query` asks for a piece of information from the device and expects an immediate reply. `Act` instructs the device to perform an action (e.g., modify some parameters of the device, or if the device is an actuator perform an action). `CreateEventID` describes a specific event that this device can produce and gives this event a name/ID. The name can be used subsequently from the `wait` command to wait on this specific event. `DisposeEventID` just disposes that name. Additionally, if a device can produce events, a task is needed to accept create/disposeEventID commands and react to wait commands that are waiting on the device's events. The task definition, and the parsing of the arguments of the four commands are defined in a custom fashion by the developer. This is where the expandability stems from, while at the same time keeping a structured form.

C.2 HW/SW variability

Even though two platforms may have the same capabilities (i.e., the same modules/services), they may rely on different hardware and/or operating system. In order to facilitate the porting process it is desirable to clearly separate the OS and HW-specific code from the fixed code and the capabilities-definition code. To achieve this we need to identify the dependencies of the code to the OS and the hardware and create abstracted wrapper functions. The wrapper functions are actually defined in separate sections of the code (i.e., different .c files) so that the developer can easily identify the points of change for a porting procedure.

From the OS we need support to create and initiate threads/tasks, and support to define, post, and pend into mailboxes/queues. Thus we create wrapper functions for these operations. We also need low-level functions to access the hardware, thus we create wrapper functions around them (these functions will depend on the specific capabilities the platform supports). Figure 7 illustrates SensorWare's code structure.

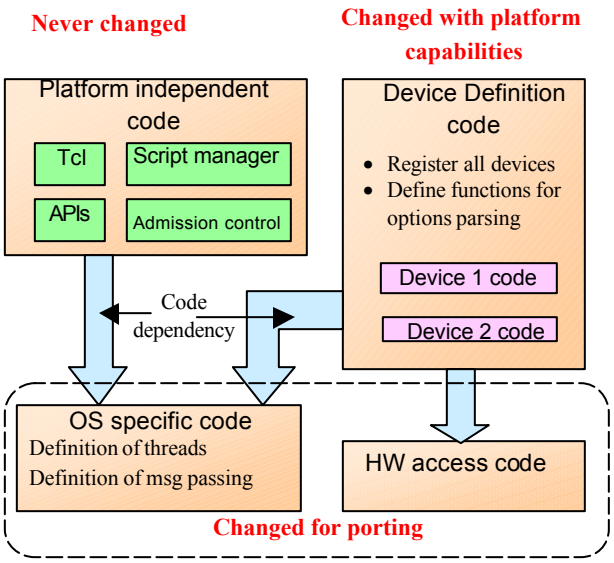


Figure 7:SensorWare code structure

V. CODE EXAMPLES

In order to make SensorWare more concrete, we will present code examples and porting details in the next two subsections. The first one involves the creation of a specific application using the SensorWare script language. The second example, present details on how to port SensorWare in a specific platform. More specifically, we will show how to define new devices and how to connect the framework with the existing OS and hardware.

A. Script example

In this subsection we will present the code for the snapshot aggregation application with multiple (static) users support. The specific problem that we are solving is to find the global maximum among current sensor node readings and report it back to the user. Furthermore, multiple users may request this maximum while the algorithm is running (i.e., time to populate the script into the network, collect and aggregate data towards the user). The users are accommodated with the minimum traffic, without the need to launch a different application/script for each user. Finding the minimum, average, or any other aggregation function, among different kinds of sensor node readings or state, can be easily achieved by trivial modification in our script. More on aggregation applications in general can be found in [3] and [4].

Before proceeding with the script code, it is beneficial to describe the internal workings of two Sensorware commands, namely "`replicate`" and "`wait`". `Replicate` (possibly) transfers the script that it was called from, to other node(s). It does not blindly pack and transmit the code and state of the script like all other active sensor approaches currently do. `Replicate` first starts with a transmission of "intention to replicate" message, carrying the name of the script and the issuing user. If the same script already exits in the other node(s) `replicate`, according to options, may choose not to transfer the code, may choose to initiate a second script of the same type in the node, or if the script has multi-user support, send an "add user" message. By default, `replicate` will send the "intention to replicate" message to avoid unnecessary code transfers, and will spawn a second script only if the requesting user is different by the existing one. Furthermore, it is assumed by default that the parent of the script (i.e., the node that spawned the script to the current node) already has the code for the script, thus does not need an "intention to replicate" message. The arguments of the `replicate` command are:

replicate [-[f] [d] [p] [m] [rc] [rs] [ru]] [node_list]
[ ] means optional
f : forced replicate, no "intention to replicate" message sent
d: duplication of script at remote node irrespective of user
p: parent not assumed to have script in memory
m: script supports multi-users. Do not spawn new script in remote node, instead send "add user" message to existing script
rc: return nodes that code was transferred
rs: return nodes that spawned new script
ru: return nodes "add user" message was sent
by default option rsru is in effect.
node_list: nodes to replicate. Leaving this field empty implies a broadcast to neighbors. Parent is excluded unless p is chosen.

It is also useful to reveal some of the details of the `wait` command. `Wait` returns when an event named in the command's arguments occurs. In order to expedite processing of the event by the subsequent scrip code, the wait command sets the following predefined variables:

event_name : the name of the occurred event. It indicates the device that caused the event and the type of the event
event_data: data returned by the event
If the event is a packet reception the following are defined and set: ***msg_sender***, ***msg_body***

Listing 1 shows the actual SensorWare script. SensorWare commands and reserved words are in boldface. Variable names are in italics. Reserved variable name are in boldface and italics. Basic Tcl knowledge is needed to follow the script, although we do explain most of the code step by step. The example is sufficient to illustrate the programming style and the use of some of the most important commands, while solving a real problem.

```
set need_reply_from [ replicate -m]
set maxvalue [ query sensor value ]
if {$need_reply_from == ""} { send $parent $maxtemp; exit }
else { set return_reply_to $parent }
set first_time 1
while {1} {
      wait anyRadioPck  // "anyRadioPck" is a predefined eventID
      if { $msg_body ==add_user } {
            if { $first_time == 1 } {
                  send $parent $msg_body
                  set first_time 0
            }
            set return_reply_to "$return_reply_to $msg_sender"
      }else {
            set maxvalue [expr {($maxvalue<$msg_body) ? $maxvalue
: $ msg_body }]
            set n [lsearch  $need_reply_from $ msg_sender]
            set need_reply_from [lreplace $need_reply_from $n $n]
      }
      foreach node $return_reply_to{
            if { ($need_reply_from=="")||($need_reply_from==$node)} {
                  send $node $maxvalue
                  set n [lsearch  $return_reply_to $node]
                set return_reply_to [lreplace $return_reply_to $n $n]
            }
      }
      if {$return_reply_to==""} {exit}
}
```

Listing 1: Multi-user aggregation code

The specific script keeps two important variables at each node: a list of nodes that replies are needed from, and

a list of nodes that replies are due. The first command tries to replicate the script to all the neighbors (except the parent), declaring that this is a multi-user script. The nodes that the script was spawned or an "add user" message was sent are returned and added to the need_reply_from variable. The second command reads the current value from the sensing device and sets the maxvalue variable with it. If there are no nodes to return a reply the script sends the maxvalue to the parent node and exits. Otherwise the parent node is added to the list return_reply_to and the big loop begins. Each time a packet is received we check if it is a data reply or an "add user" message and modify our lists and maxvalue accordingly. To graphically see how this algorithm works, refer to [4].

The script is its raw form is 882 bytes. If reserved words and variable names are compressed, the script becomes **277** bytes. If furthermore, we compress this intermediate form with gzip, we end up with **209** bytes. This is a compact description for this non-trivial algorithm. An equivalent SQTL script has a size in the order of 1000 bytes (based on the simpler algorithm of aggregation for a single user and without replication checking). Building the same algorithm in Maté was proven impossible due to its limited heap and stack sizes. There was not enough space to hold the need_reply_from and return_reply_to lists. Even with a larger memory space though, Maté's stack based architecture and lack of higher-level services results in code of many instructions even for simple tasks. As stated earlier, Maté's restrictions are a design choice, coming from the desire to support the restrictive underlying platform. Finally, C code is written for this algorithm, with external references to SensorWare functions. The compiled native code has a size of **764** bytes (without including the size of SensorWare functions called from within the native code).

B. *Porting SensorWare to a platform*

In this subsection we will present some of the issues while porting SensorWare to a platform. We consider our iPAQ-based prototype as the testbed. A full description of the platform can be found in section VI.A. Here it is sufficient to know that the node has one radio and one sensing device, and that the underlying OS is Linux.

First, we should add the proper capabilities to SensorWare by creating a virtual device for the sensing device (the radio has a virtual device by default). This means name and register the device by calling the function:

```
create_device(char* name, int (*query)(), int (*act)(), void*
(*createEventID)(), int (*disposeEventID)(), void* (*task)() )
```

As it can be seen by the declaration of the create_device function we need to define the four functions to parse the arguments of the four standard interface commands, plus a function to be executed by the thread/ task of the device. Not going any further into the definition of these functions, we are sufficed to say that they are very similar to the radio device functions.

The next step is to define the OS-specific code. More precisely, have the ability to create threads and use mailboxes/queues. For the definition and creation of threads we use the pthreads (i.e., posix threads) provided by Linux. Even though mailboxes are available in Linux, we chose to construct our own structures using semaphores. Finally, the hardware-specific code is directly provided by the Linux's device drivers.

VI. IMPLEMENTATION

Some active sensor frameworks choose to evaluate their performance by showing their expressiveness. They create a distributed algorithm for a particular application and compare it against a more centralized approach (usually a distributed database approach). We believe that the energy savings from such comparisons are evident for *any* active sensor framework and do not add value to the investigation and evaluation of the framework. To evaluate SensorWare we chose to implement it and measure the overheads we are paying for dynamic programmability. How much memory do SensorWare and its components occupy? How much delay is introduced by various SensorWare operations? How much slower and consequently how much more energy-consuming is SensorWare compared to native code approaches? These questions are answered in the following subsections. We begin by a description of the implementation platform.

A. *Platform description*

The prototype platform used in the implementation and evaluation of SensorWare was built around the iPAQ 3670 [15]. The iPAQ has an Intel StrongARM 1110 rev 8 32 bit RISC processor, running at 206Mhz. The flash memory size is 16Mbytes and the RAM memory size 64Mbytes. The OS installed is a familiar v0.5 Linux StrongARM port [9], kernel version 2.4.18-rmk3. The compiler used, is the gcc cross-compiler. A wavelan card [28] is used as the radio device and a Honeywell HMR-2300 Magnetometer [13] as the sensing device.

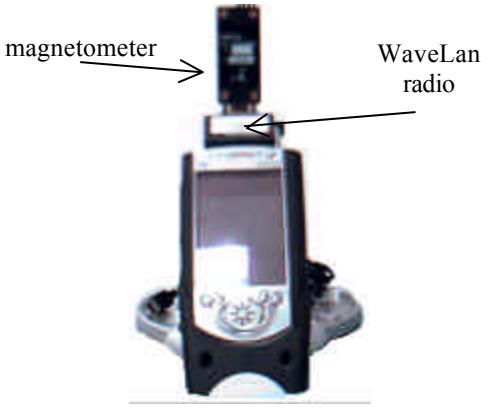


Figure 8: The implementation platform

SensorWare is also ported into the Rockwell WINS nodes [24] that also have a StrongARM processor, but only 1Mbyte of flash memory. Both eCos [6] and microC/OS-II [18] were used as operating systems for these nodes.

B. *Memory size measurements*

The first question to answer is how much size does the whole framework occupy. Figure 9 shows that the total size is 179Kbytes and it is consisted of 74Kbytes of Linux specific code (e.g., kernel, libraries), 74Kbytes of a stripped down Tcl core called tinyTcl, 22Kbytes of SensorWare code and 8Kbytes of platform dependent code (i.e., functions to access the hardware). The bottom part of the figure shows the breakdown of the SensorWare core part into smaller parts.

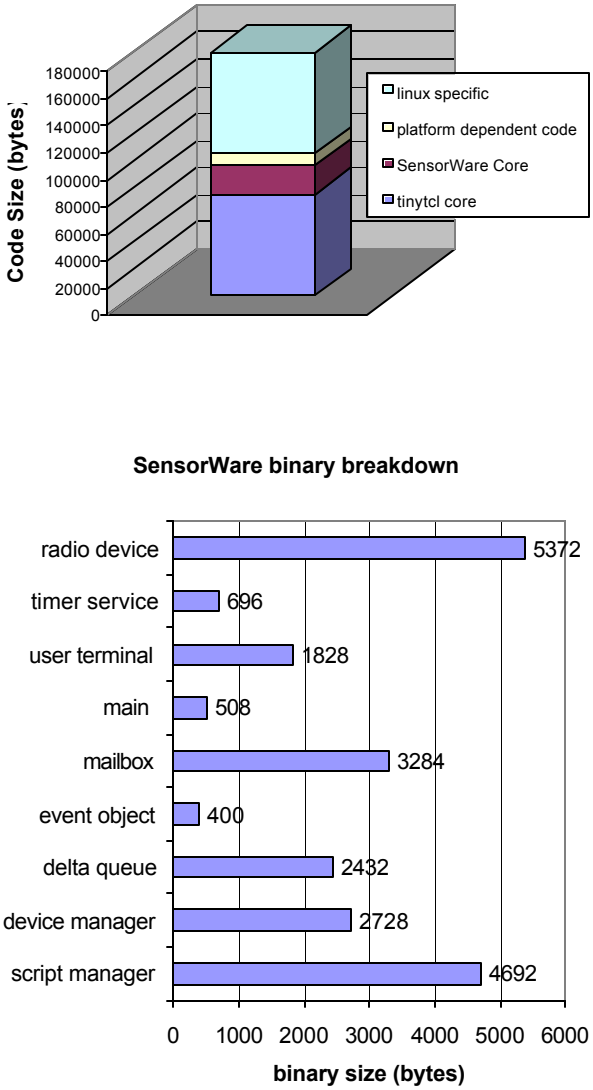


Figure 9: Code size breakdown

C. *Delay measurements*

The next question to answer is how long do different basic commands need to execute. We measured each command individually 100 times under the same basic conditions (only one script executing) and derived an average and standard deviation for the delay. Most commands exhibited negligible variance. All the commands, except the ones that used the radio and the one that spawned a 50byte script, have an execution time less than 0.3msec.

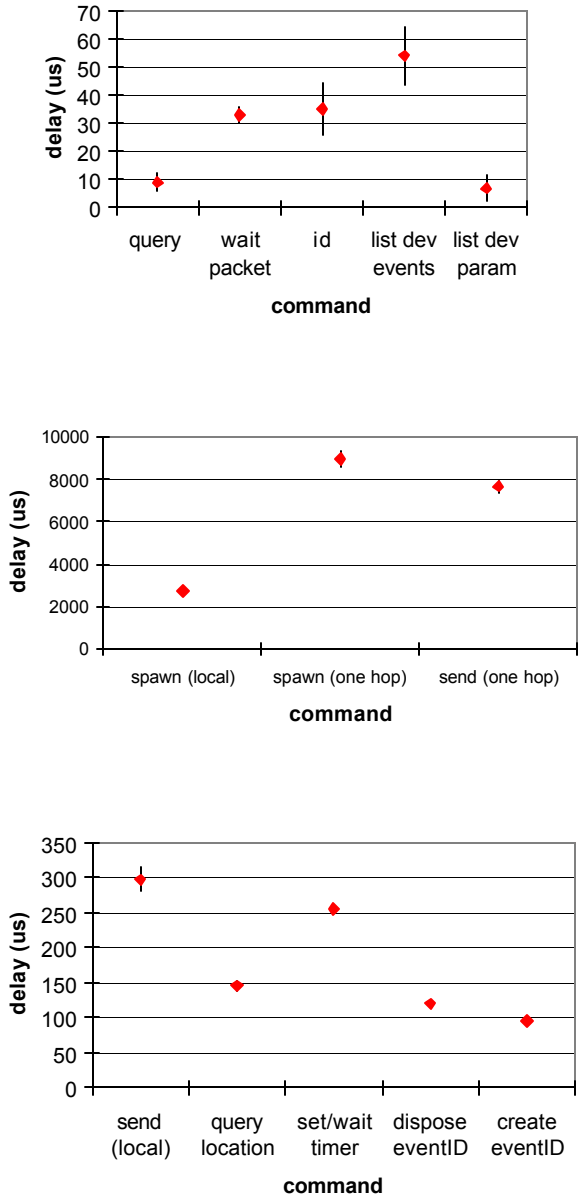


Figure 10: Execution times of SensorWare commands

The top graph of the figure 10, shows commands with less than 0.06msec delay. The last two commands that return some part of the device's state are internal to SensorWare and not exported for script use. The middle

graph shows the most time consuming commands. The first one spawns a 50 byte script locally. The other two commands use the radio to spawn a script in a neighboring node and send a message in a neighboring node. The delay for achieve these two operations is dominated by the radio transmission time. Note that the send command and some operation modes of the spawn command, do not wait for the whole operation to finish, instead they return as soon as they hand off the task to the radio device. In the graph, the total operation time is shown. The bottom graph of figure 10, shows yet another set of delays. Of particular interest is the set/wait timer delay. For this instance, we measure the delay to set a zero-valued timer and wait for its expiration. In essence we are measuring the overhead of real-time measurements in scripts. The overhead is 0.25msec with very small variation. Thus, we can internally subtract this number each time a script sets a timer, in order to measure the true desired time.

In order to acquire all delay measurements we used the gettimeofday() system call. This function is based on the timer count register found in the StrongARM processor. The accuracy of this method is measured to be 1μsec.

D. Energy measurements - related tradeoffs

Finally, we are interested in knowing the energy overhead from the interpreted nature of SensorWare. For that purpose we compare the interpreted version of the script presented in section V.A., with a compiled native code version of the same algorithm. The native version uses the services that SensorWare provides by directly calling the appropriate functions. Since most of the work inside a script is done by the SensorWare commands and services (which are implemented in native code) we do not expect a significant change when we resort to fully native code. Indeed, we measured an 8% speedup of the native code compared to the interpreted code. We acquired this number by measuring the total execution times of both codes, and *excluding* time periods when the code was accessing the radio, or was waiting for events to occur. Essentially, the time we measured, was the non-idle CPU time. This time is linearly coupled with the energy spent on the CPU, assuming that we have a mechanism to shut down the CPU during idle time. Thus a reduction of 8% in the non-idle time, directly translates to a reduction of 8% in CPU-energy spent.

As we already mentioned in section V.A, the script has a final compressed size of 209 bytes, while the native code has a size of 764 bytes. So even if the native version executes faster (and potentially consumes less energy, by allowing to shut down the CPU during idle time), there is an energy overhead related to its transmission. The wavelan radio in typical operation would spent 0.47mJ to transmit the script, and 1.10mJ to transmit the native code (including the MAC overhead). Thus, the energy difference between the two transmissions is 0.63mJ. The typical power for the StrongARM is 230mW, so 0.63mJ are spent in 2.7msec. From these numbers we deduce that if the native code uses StrongARM for 2.7msec less than the interpreted code then its initial transmission energy overhead is balanced. For the particular algorithm that we tested, 8% speedup is translated into 1.2msec gain in absolute numbers. So for the particular algorithm transmitting and executing native code is not beneficial overall. For applications with heavier computation workload it might be desirable, from an energy viewpoint, to transmit and execute native code. Note though that we would sacrifice the portability of the code in several platforms, and most importantly we would sacrifice the code safety offered by the scripts (refer to [2] for more information on scripts code safety). Furthermore, most sensor node platforms have a much slower radio than wavelan, which in turn means that they spend more energy to transmit the same amount of bytes, changing the tradeoff points. In conclusion, for most sensor node platforms, one would have to have a very computation-intensive algorithm to prefer the native code over SensorWare scripts.

VII. CONCLUSIONS

In this paper we argue that the development of a framework based on a scripting abstraction where the scripts are mobile, will help bring many desired properties in sensor networks. It will make the sensor networks programmable and open to external users and systems, keeping at the same time the efficiency that distributed proactive algorithms have. We explain the framework's architecture and present code examples. Through our implementation we are able to measure the time and energy overheads that we are paying for programmability and explore some part of the solution space for sensor node run-time environment abstractions.

VIII. REFERENCES

[1] P. Bonnet, J. Gehrke, and P. Seshadri, " Querying the Physical World", IEEE Personal Communications, October 2000.

[2] Withheld to preserve authors anonymity.

[3] Withheld to preserve authors anonymity.

[4] Withheld to preserve authors anonymity.

[5] L. Clare, G. Pottie, J.R. Agre, "Self-Organizing Distributed Sensor Networks", Proceedings of SPIE conference on Unattended Ground Sensor Technologies and Applications, pp. 229-237, April 1999.

[6] eCos: Embedded Configurable Operating System, http://sources.redhat.com/ecos/

[7] D.Estrin, R.Govindan, J.Heidemann (Editors), "Embedding the Internet", Communications of the ACM. Vol. 43, no 5, pp. 38-41, May 2000.

[8] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", ACM Mobicom Conference, Seattle, WA, August 1999.

[9] Familiar Project, "http://familiar.handhelds.org".

[10] M. Hicks, P. Kakkar, J. Moore, C. Gunter and S. Nettles, "PLAN: A Packet Language for Active Networks", Proceedings of the International Conference on Functional Programming (ICFP '98), 1998.

[11] J. Hill and D. Culler, "A wireless embedded sensor architecture for system-level optimization", Intel Research IRB-TR-02-00N, 2002.
[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System Architecture Directions for Networked Sensors", Proceedings of ASPLOS-IX, November 2000 Cambridge, MA, USA.
[13] Honeywell HMR-2300 Magnetometer, http://www.ssec.honeywell.com.
[14] T. Imielinski and S Goel, "DataSpace: Querying and monitoring deeply networked collections in physical space", IEEE Personal Communications, Oct. 2000.
[15] iPAQ 3670, http://thenew.hp.com/.
[16] C. Jaikaeo, C. Srisathapornphat, and C. Shen, "Querying and Tasking of Sensor Networks", SPIE's 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Control (Digitization of the Battlespace V), Orlando, Florida, April 26-27, 2000.
[17] D. Kotz, R. Gray, "Mobile Agents and the Future of the Internet", in ACM Operating Systems Review, 33(3), 1999.
[18] J. Labrosse, " MicroC/OS-II: The Real Time Kernel", CMP Books, November 1998.
[19] P. Levis, D. Culler, "Maté: A Tiny Virtual Machine for Sensor Networks." Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X), October 5-9 2002.
[20] J. K. Ousterhout, "Scripting: higher level programming for the 21st Century", Computer, vol.31, (no.3), IEEE Comput. Soc, March 1998. p.23-30.
[21] J. K. Ousterhout, "Tcl and the Tk toolkit", Addison-Wesley, 1994.
[22] G.J. Pottie and W.J. Kaiser, "Wireless Integrated Network Sensors", Communications of the ACM. Vol. 43, no 5. May 2000.
[23] Reactive Sensor Networks, http://strange.arl.psu.edu/ RSN/
[24] Rockwell WINS nodes, http://wins.rsc.rockwell.com/
[25] SenseIT program, http://www.darpa.mil/ito/research/sensit/index.html
[26] C. Srisathapornphat, C. Jaikaeo, and C. Shen, "Sensor Information Networking Architecture", International Workshop on Pervasive Computing (IWPC'00), Toronto, Canada, August 21-24, 2000.
[27] D. Tennenhouse, "Proactive Computing", Communications of the ACM. Vol. 43, no 5, pp.43-50, May 2000.
[28] Wavelan card, "http://www.orinocowireless.com"

APPENDIX

A. The SensorWare Language

SensorWare supports Tcl syntax and the following 41 Tcl commands: append, array, break, case, catch, concat, continue, error, eval, expr, for, foreach, format, global, if, incr, info, join, lappend, lindex, linsert, list, llength, lrange, lreplace, lsearch, lsort, proc, regexp, regsub, rename, return, scan, set, split, string, trace, unset, uplevel, upvar, while.

There are 11 other commands defined by SensorWare that essentially abstract the node's run-time environment. They are:

spawn [-[f] [d] [p] [m] [rc] [rs] [ru]] [<node_list>] <code> [<variable_list>]

replicate [-[f] [d] [p] [m] [rc] [rs] [ru]] [<node_list>] [<variables_list>]

migrate [-[f] [d] [p] [m] [rc] [rs] [ru]] [<node_list>] [<variables_list>]

send [<node_id>][:<script_name>] <message>

setTimer <timer_name> <value>

disposeTimer <timer_name>

query <device_name> [var_arg...]

act <device_name> [var_arg...]

createEventID <device_name> <eventID> [var_arg...]

disposeEventID <device_name> <eventID>

wait <event_name>...

Legend: [ ] indicates optional, < > indicates a variable (either a Tcl variable or an SensorWare variable such as an eventID or a timer name), the suffix "_list" in variable names indicates that the variable is a list (i.e., zero or more elements). The symbol "var_arg ..." indicates variable arguments. The modifier "..." indicates a list of arguments of the preceding argument type.

There are 6 reserved Tcl variable names. These are: parent, neighbors, event_name, event_data, msg_sender, msg_body.

There are 7 reserved words used as arguments in some commands. By reserving words for commonly used features we compact the scripts further. These are: anyRadioPck, anyTimer, add_user, sensor, value, radio, timer.